MEMORANDUM
RM-3337-PR
FEBRUARY 1963

A GUIDE TO THE GENERAL PROBLEM-SOLVER PROGRAM GPS-2-2

Allen Newell

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-3337-PR
FEBRUARY 1963

A GUIDE TO THE GENERAL PROBLEM-SOLVER PROGRAM GPS-2-2

Allen Newell

This research is sponsored by the United States Air Force under Project RAND—contract No. AF 49(638)-700 monitored by the Directorate of Development Planning, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force. Permission to quote from or reproduce portions of this Memorandum must be obtained from The RAND Corporation.

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA

PREFACE

This Memorandum provides a detailed account of the internal structure of a problem-solving program, the General Problem-Solver (GPS). This program in its various versions has been one central part of work at RAND on artificial intelligence and simulation of cognitive processes during the past five years. Although GPS has been reported on many times, there has never been a completely adequate account of its detailed structure. This Memorandum attempts to fill this gap.

This guide will be of use only to those who are deeply and technically concerned with the problems of programming complex systems. It is essentially a reference document which provides a level of description which is normally unavailable in the field of complex programs.

The general field of artificial intelligence and information processing psychology, to which this Memorandum is contributory, aims at understanding the complex information processes that underlie man's ability to solve problems, learn, adapt, and create. From a scientific viewpoint, such activities are intrinsically worthwhile; from an applied viewpoint they form the essential basis for increasing the sophistication and eventual effectiveness of our large command and control systems.

GPS in its various forms and guises is the joint work of J. C. Shaw of RAND, H. A. Simon, and the author. The latter two are members of the faculty of the Carnegie Institute of Technology, and consultants to The RAND Corporation.

SUMMARY

The General Problem-Solver (GPS) is a computer program being used for explorations into both the general mechanisms involved in problem-solving and the way humans solve problems. The program has existed in several versions since it was first developed in 1957. This Memorandum is a guide to the detailed structure of one of the versions, GPS-2-2. It assumes a substantial knowledge of IPL-V, the programming language in which GPS is written, and a general knowledge of GPS as it has been described in the published literature. It is also meant to be used in conjunction with an assembly listing of the program, but can be used alone.

After the Introduction, Sec. II gives the gross topography of the program. It also includes a run-through of a simple problem to put the parts in context. Section III discusses the various data structures used in GPS: goals, expressions, derivation lists, operators, location programs, and differences. Section IV is devoted to a detailed description of the subroutine hierarchy, working from the top executive down through the technique of method interpretation to a consideration of each method and method segment. Section V describes the Experimenter; i.e., the embedding program used to put GPS into operation and to output selected aspects of its performance. Section VI takes up the information provided for each task environment. For GPS-2-2 these are Logic, and Missionaries and Cannibals. In addition there is a description of how a new task environment might be added to GPS. Four appendices provide additional specific data on the program.

CONTENTS

I. INTRODUCTION

The General Problem-Solver (GPS) is a computer program being used for explorations both into the general mechanisms involved in problem-solving and into the way humans solve problems. As its name indicates, there is both an aspiration that GPS should be capable of handling a wide range of tasks and the fact the GPS's organization is task independent in many respects.

GPS grew out of The Logic Theory Machine,[1-3] a program for proving theorems in the sentential calculus of Whitehead and Russell. The first version, called GPS-1, was coded in IPL-IV for JOHNNIAC.[4] The most complete description of GPS existing in the published literature is the "Report on a General Problem-solving Program for a Computer,"[5] which gives only the highest level organization. A discussion of some organizational issues arising in GPS will be found in "Some Problems of Basic Organization in Problem-solving Programs."[6] GPS has been discussed in several other papers in connection with its use as a simulation of human thought[7-11] and in an investigation of learning.[12] A recent paper[13] also discusses the first steps in getting GPS to program by constructing an independent, GPS-like program called the Heuristic Coder.

GPS rapidly outgrew the small storage capacity of JOHNNIAC (4096 words), and was recoded in IPL-V to run on the 704-709-7090 series machines, which have 32,576 words of fast storage. The new program was called GPS-2-1. Functionally it was almost identical to GPS-1, but substantial organizational changes were made. The change to GPS-2-2 involved somewhat smaller organizational changes, but required a separate designation, since both versions were running at the same time. This document is a description of the structure of GPS-2-2. GPS-2-1 is not

separately documented and is no longer a functioning program. Additional versions, GPS-2-3 and GPS-2-4, now exist. They involve more substantial organizational changes from GPS-2-2, and will be documented separately.

This document is a guide to someone trying to understand the GPS program in detail; it is not written as a general introduction. It assumes knowledge of IPL-V[15] and the published general descriptions of GPS. Thus, the user of this guide should already understand that GPS uses goals of three types in a recursive way to build up a hierarchical goal tree for the problem at hand; and he should understand in a general way the nature of the methods that generate this tree and the devices that are used to prune the tree. He will find in this guide numerous additional mechanisms that are unmentioned in the published papers.

IPL-V is written in a vertical format with specific fields assigned to various parts of the IPL words. We will adopt a convention here that will allow us to write IPL code without specifically assigning fields on the page. We use a slash (/) to separate NAME from PQ SYMB and a period (.) to separate PQ SYMB from LINK. Thus the following IPL program would be transcribed as shown below:

NAME	PQ	SYMB	LINK
P7	10	L5	
		P4	
	11	WO	
		J2	
	70	9-1	9-2
9-2		P5	
		P6	P8
9-1		P9	0

```
P7/  10L5
       P4
     11WO
       J2
     709-1.9-2
9-2/   P5
       P6.P8
9-1/   P9.0
```

We will use an equals sign (=) to indicate an integer data term; e.g., 9-1=5 means that 9-1 is the name of data term integer 5. (We will have no occasion to use the other types of data terms.)

II. GENERAL STRUCTURE OF THE PROGRAM

REGIONS

The total program is divided into several parts: the Experimenter, the GPS-Core, and the various specialized parts for each task environment. Each part uses symbols from different regions for its routines and data. One rough picture of the total program is obtained by giving a schematic division of the 32K store into the separate parts, showing the different regions and their functions. The amount of space devoted to a part and the number of separate entities is necessarily approximate, since the program is under continual modification. It also includes numerous alternative versions of routines and lists.

GPS-Core: 4700 words

- A: General attributes (50) (one word per attribute)
- G: Goal attributes (50) (one word per attribute)
- P: Basic routines (70) (1200 words)
- Q: Y cell routines (100) (2500 words)
- R: Top level routines (10) (500 words)
- K: Constants and lists (60) (300 words)
- Y: Local context working cells (100)

Logic Task Environment: 2100 words

- F: Routines (20) (900 words)
- C: Constants and lists (50) (400 words)
- D: Differences (50)
- B: Operators and objects (80) (800 words)

Missionaries and Cannibals Task Environment: 900 words

- M: Routines (10) (600 words)
- M: Data (20) (300 words)

Experimenter: 1100 words

- E: Routines (50) (900 words)
- L: Lists (30) (100 words)
- Z: Cells and constants (100)

Additional Basic System: 200 words

I: Routines (10) (50 words)
S: Signals (100) (one word per signal)
N: Integers (100)

IPL-V System: 7000 words

H: Basic communication cells (10)
J: Primitive routines (200)
W: Working cells (30)

Working Space: 16,000 (less after set up, conversion, etc.)
Goals run about 100-150 words per goal
Expressions (in logic) run about 30 words per expression

GPS-CORE

The data structures that GPS uses are expressions, which describe the objects GPS wishes to manipulate; and goals, which describe the situations GPS wishes to obtain. (There are also a few miscellaneous structures.) These are described by numerous attributes. G-symbols (e.g., G1, G25) are used for attributes that are peculiar to goals; A-symbols are used for all other attributes. All A-symbols and G-symbols define routines of identical form--for example: A1/ 10A1.J10. Thus executing A1 on the name of an expression will retrieve the value of attribute A1 on the description list of that expression. The situation is similar with the G-symbols, except they check to see if the goal is stored on auxiliary storage.

GPS is always in the context of attempting a single specific goal. The goals form a hierarchical network, so that one may visualize the program in operation as located at some one node of this network. Depending on the result of problem-solving activity on this current goal, the program will move to another goal; e.g., back up to the supergoal, down to a newly created subgoal, and so on.

The Y-cells hold the immediate context. That is, they hold the information pertinent to the current goal that is being attempted. Each Y-cell has a specific function. For example, Y2 holds the name of the current goal, Y3 holds its type, and so on. Each Y-cell holds only a single symbol, so that when one says, for example, "the goal in Y7" or "the expression in Y13" one means the list structure whose name is in the cell Y7 or Y13, respectively. (Y-cells are occasionally pushed down on a temporary basis within a single Q-routine, but this is a local matter, not within the cognizance of the system conventions.) Thus, the gross action of the program is to get into the context of a goal by setting the Y-cells appropriately; to engage in some problem-solving activity, working in and out of the Y-cells; to record the information that should be kept permanently in the goal structure; and to leave this goal context for another one.

The routines of the core are divided according to their relation to the Y-cells. At the top level there are R-routines. These are independent of the Y's and follow a special set of coding conventions. Next come the Q-routines. These routines take their inputs from the Y-cells and put their outputs back in the Y-cells. Thus, the R-routines use the Q-routines in order to accomplish all their actions. Finally, there are the P-routines. These are general purpose routines that take their inputs from HO and put their outputs in HO. They know nothing of the Y-cells either. (A major purpose of this division is to guarantee that Y-cells are safe over P-routines.)

TASK ENVIRONMENT

Basic to the current version of GPS is the assumption that problems or tasks can be grouped into large classes which are homogeneous with respect to the particular facts,

heuristics, operations, etc., required to solve them. Thus, there is a collection of particulars that make up "knowing about chess" or "knowing about symbolic logic"; if these are known, then many problems about chess (or symbolic logic) can be posed and attempted. GPS-Core makes no reference to such particulars. It knows only about "objects" and "operators" in the abstract; e.g., that there are differences between objects, that two objects can be put into correspondence, and so on. The additional program and data needed to complete GPS so that it can work on tasks of a given class is called a task environment part. The symbolic logic task environment part, for example, consists of routines that accomplish input and output conversions; routines that compare two expressions to determine what differences hold; routines that perform similarity tests and identity tests; and data structures for the table of connections, the operators, and logic expressions. The Missionaries and Cannibals task environment, the other environment that exists in GPS-2-2 in completed form, is similar in structure. Its principal addition is a routine for accomplishing the basic operators (M22), since these are not conveniently expressible as forms of the same kind used in symbolic logic.

It is assumed that the routines of a task environment know about the Y-cells and accomplish their functions by working directly into and out of Y-cells. They may use the P-routines as subprocesses, but may not use either the Q- or R-routines.

The information for each task environment is given by a list (K70 for O. K. Moore symbolic logic, M19 for Missionaries and Cannibals). GPS always works in the context of a single task environment (TE), given in Y4. There is a routine, Q79, which changes TE's. To be in

context for a TE means to have all the routines and data for that TE available (currently localized to Y50-Y79, K80-K89, and Z80-Z89). A TE list is formed as a list of pairs: the name of the cell that should hold a given type of information followed by the symbol it should hold for this TE. Q79 accomplishes the function of changing to a new TE, including blanking out all the cells from the old TE (by putting K92 in the cell), so that old routines and data will not be spuriously used.

EXPERIMENTER

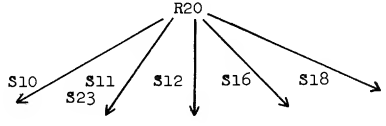
Besides the problem-solver, which consists of GPS-Core plus the TE parts, there is another part of the program, called the Experimenter, whose function is to handle input and output conversions; to make the initial setup; to present GPS with the problems we wish it to attempt; to provide GPS with any appropriate "external environment" (such as the autonomous play of an opponent); and to monitor the activity of GPS for any debugging or performance data. The Experimenter has its own routines (E-routines) and its own lists (L's) and its own cells and constants (Z's). Insofar as these occur in P-, Q-, and R-routines, they indicate monitor and output functions and have nothing to do with the problem-solving activity.

ADDITIONS TO THE BASIC SYSTEM

GPS is coded in IPL-V and uses the basic set of J-routines already available. The only additions to this are a universal set of symbols for positive integers (Nx for integer x); a few scattered P-routines, which accomplish basic lists processes (such as P60/ J74.J136, copy and make local); a couple of routines (I20, I21) for handling the assembly and correction procedures; and the signal system. This latter is sufficiently important to be described in detail.

SIGNAL SYSTEM

The purpose of the signal system is to allow a symbolized multi-way branch as a basic coding operation. Imagine a routine, say R20, accomplishing some function and leaving in a special cell (the signal cell) one of several symbols, say S10, S11, S12, S16, S18, or S23. Each of these indicates that some particular generalized outcome has occurred. We now want to transfer to different routines depending on which outcome happened. As a flow diagram, we might write:



In IPL-V terms we can think of following the execution of R20 with a list of pairs, the first symbol giving the signal (S-symbol) and the second giving the location to transfer to:

```
9-1/  R20.9-2
9-2/  I1
      S10
      9-10
      S11
      9-11
      S12
      9-12
      S16
      9-13
      S18
      9-14
      S23
      915.0
```

We can visualize the execution of this structure proceeding as follows. Instruction 9-1 is executed. This leads to R20 being executed, resulting in a signal, say S12, being put in the signal cell. Having finished R20,

the next instruction to be executed is 9-2, which leads to the execution of I1. I1 now goes to H1, the current instruction address list, and recovers the symbol 9-2. It then searches down 9-2 looking for the symbol that matches the symbol in the signal cell (S12). This search is essentially J10: find the value of an attribute on a description list. In this case the value of S12 is the symbol 9-12. I1 removes 9-2 from H1, since the list 9-2 should not be executed as a string of instructions, and puts 9-12 into H1 in such a way that the next instruction that is picked up for execution is 9-12. (The routines that manipulate H1 [I9 and I19] must really be considered additions to the IPL-V basic system, since they imply detailed knowledge of how the IPL-V interpreter works.) If the signal is not found in the list, a special signal, S9, is used to stand for "in all other cases," and a search is made to see if S9 is on the list. If S9 is not on the list, then I1 behaves like J0.0--that is, the routine (at this level) terminates.

All S-symbols are signals. The signal cell is Y1 and all S-symbols are defined as routines which put their name into Y1:

```
S12/  10S12
      20Y1  .0
```

It is sometimes desirable to take a multi-way branch on some other class of symbols than the signals. Thus, I2 is a routine analogous to I1, but taking its symbol from Y18, which contains the current difference symbol. This permits a discrimination on the difference that is being considered. Similarly I3 takes its symbol from Y3, which contains the goal type, and I4 takes its symbol from Y85, which contains the expression type.

Besides I1, I2, etc., the routines I11, I12, etc., are also defined. I11 is identical to I1, except that in

I1 the signal is recorded for output (via the monitor routine, E70, in Z92) whereas in I11, it is not. The situation is similar for the others.

A TOUR THROUGH A SIMPLE PROBLEM

To provide an overview of the operation of the program, its behavior on a simple problem will be described. All the information in this subsection is described in more detail elsewhere, so that only the single thread that GPS follows need be outlined.

The problem is C36, Transform R.(-PIQ) into (QVP).R. A trace of the program's behavior is given in Appendix A. The first page is a list of specifications; the only part that concerns us here is the task environment part, specified to be K70, and the task, specified to be C36.

The program starts at E2. This is the top executive of the experimenter and oversees the conversion of all the inputs into internal form (including the assignment of names like R1 to operators and L1 to objects). By the time the trace begins to print, all the conversion of goals, operators, and objects has been completed and E2 has fired R2, which is the top executive of GPS-Core. The number at the far right shows that 35,592 IPL cycles have already gone by. R2 prints out the two expressions, the goal expression, and sets up three derivation lists. These lists hold the names of expressions that have been derived from a common source. Thus, list 28 now holds L1. As soon as some operator is applied to L1 to produce a new expression, then the name of this expression is put on 28. Adding to 28 is essentially working forward; adding to 29 is working backward (not done in this run). All the operators are on list 30, and any new operators that are generated (not done in this run) would be put on list 30 as well. R2 also sets a limit to the complexity of the expressions that GPS will consider (which does not affect behavior in this run).

At this point, R2 executes the main problem-solving executive, R10. From here on the trace gives a blow-by-blow account of all the decisions that are made. The lines of symbols that run across the page are the signals that occur at each point in the higher programs and that are used to control the transfers (see the earlier section on the signal system). The names of the R-routines are also recorded in the "signal line" to make it easier to keep track of what decisions are occurring. In the appendices, along with the run, is a series of flow diagrams for these higher routines. They should be consulted as we go through the behavior.

At the moment when R10 takes over, Goal 1 (C36) is the current goal (it is also the only goal). Thus, its name is in cell Y2, and as long as we are working on it directly, various information about it will occupy other Y-cells. Most of these are blank at the moment, since nothing has happened yet.

Consulting the flow diagram for R10 we see that we enter at Q1. Q1 tests if the "external limits" are violated—either too many cycles or too great a depth in the goal tree. The signal (in Y1) was originally set by R2 to be S50; if either of these limits had been violated Q1 would have changed the signal (to either S72 or S74). What we observe in the signal line of the trace is an S50 right after "R10." This is the signal that existed after Q1; thus no limits were violated and the next Q-routine to be executed is Q2.

Q2 finds the next method. There is a list of methods associated with each goal, consisting of the method name followed by a status symbol, which shows whether the method has been used with the goal, whether it can still be used, etc. In this case, of course, no methods have been tried at all and method K40 is chosen. Its status (S50) is made the signal, so that on the trace we see a second S50 just before we go into R11.

A method is not a program; instead it is a list of method-segments. Each segment is a routine. R11 is the program that executes these segments and interprets the signals that are sent back from them. Method K40 is the method that matches two expressions against each other and sets up subgoals to reduce the difference between them. In Fig. 1* we give a flow diagram that is similar to the ones in the published papers but containing more detail.

R11 first detects that the signal is S50, which indicates that it is to perform the first segment of the method, R30. This segment sets the two expressions to be matched, L1 and L0, into the Y-cells and then calls on the match routine proper (R20) to match them. R20 reads the signal, S19, which tells it that it is at the beginning of a match. This leads it (see R20's flow diagram) to Q47, which checks that the match is between two objects (which it is), rather than between, say, an object and a set of objects. No difference being found, the output is S20, which means, "I have a point of correspondence between two objects which needs comparison," and leads to Q20. The total match proceeds by a series of comparisons as the various parts of the two expressions are brought into correspondence. At this point, the total expressions are being compared; i.e., the connective (here both dot), the sign of the total expressions (here both positive), whether both expressions have the same letters (here both have one occurrence each of P, Q, and R), and whether their arrangement is the same. On this last a difference is found, in that the left of L1 has R, whereas no R occurs on the left of L0 but does occur on its right, and analogously with P and Q. Thus, R20 finds that a difference in position,

*All figures, in addition to appearing in the text, are reproduced together in Appendix D.

Transform A into B

Method K40

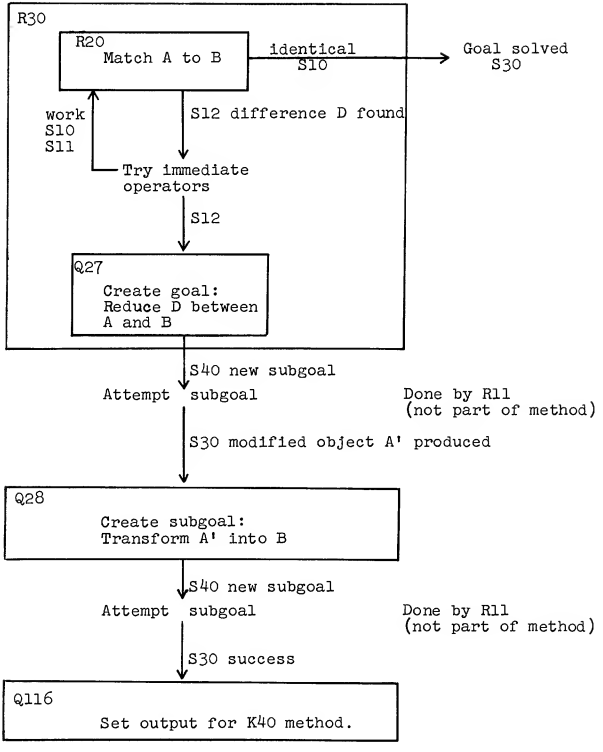


Fig. 1 Rough Flow Diagram for K40 Method.

D9, exists at the top level of the expressions. This causes the signal to be set to S12, and since S12 is not in the discrimination list of R20, R20 quits at this point and control returns to R30.

The response of R30 to S12 is not to set up a subgoal, but to see if there are any "immediate operators" that might take care of the difference right away. An immediate operator is pragmatically defined as a course of action that is guaranteed to remove the difference without further caution. Examples from this logic task are substitution and the elimination of double negation signs (such as --P into P). To this end GPS discriminates on the difference symbol (D9) which is in Y18 (and shows in the trace). This occurs twice, once for GPS-Core immediate operators and once for logic operators, but with no success. Hence R30 is led to Q27, which is the routine for creating the subgoal of reducing the position difference between L1 and L0. All the information for creating this goal is sitting in the Y-cells—the names of the objects, the difference symbols, the names of other goals to which this new goal should be linked, and so on. This first segment of the K40 method ends at this point with Q27 setting the signal to be S40—there is a new subgoal created. However, before Q27 could set S40 it had to check whether this new goal was like any other goal in the memory. In this case there was only Goal 1 to worry about, and Goal 2 was indeed found to be a new one.

At this point we are back in R11, having performed the first segment. The decision to work on the subgoal is not part of the method, but is made by R11 using the routine in Y92 (which happens to be Q74, as can be seen from the initial sheet of specifications). The result of this goal evaluation is S8, which means "undefined," and can be interpreted as saying that no goals could be found

against which to compare Goal 2. R11 interprets this to imply that Goal 2 should be tried, so it is led to execute the problem-solving executive in Y90 (which is R10) on Goal 2. This requires, first of all, that GPS get out of the context of Goal 1 and into that of Goal 2; Q81, which immediately precedes 1Y90 in R11, accomplishes this. Later on, when this attempt at Goal 2 is over, Q82 will perform the task of bringing GPS back into the context of Goal 1. This change of goal contexts involves changing the contents of the Y-cells.

The cycle now starts over with Goal 2. R10 first checks the external limits (Q1) and gets S50; it then obtains a method and finds an untried one (S50); it then goes to R11 to carry out this method. This method, K42, is given in Fig. 2. Its first segment, which is now executed by R11, is R32. It consists of finding a relevant operator to apply. The initial selection is done from the table of connections, where the difference (here D9) is used to select a sublist of relevant operators. These are subjected to some additional tests. First, they should not have been used before. There is a list of used operators on the goal against which to check; at this stage, of course, none have been used. Then each operator is subjected to a preliminary test of feasibility. This test requires, among other things, that the connectives of the operator and the expression agree. L1 has a dot main connective, so that one form of R1 (AVB => BVA) is rejected but the second form (A.B => B.A) is accepted. This can all be seen in the signal line of the trace, where the S69 shows that we are dealing with form operators (as opposed to various other kinds of operators that are possible); the first S1 S2 shows the selection of the AVB rule as untried (S1 = OK) and its rejection as infeasible (S2 = reject); and the next S1 S1 shows the selection

Reduce D from A to B

Method K42

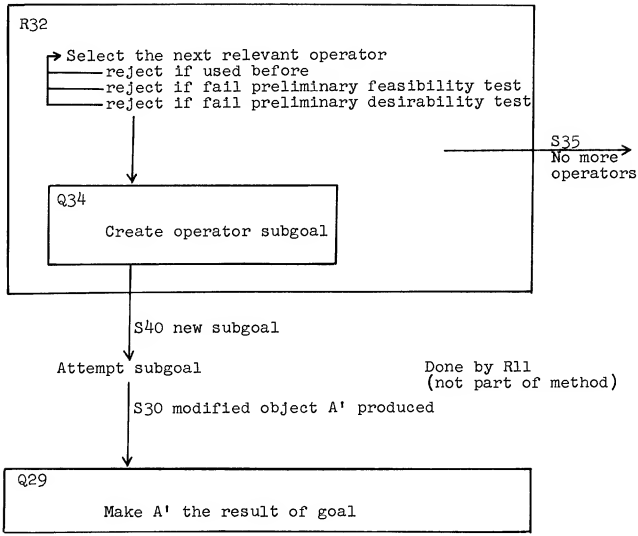


Fig. 2 Rough Flow Diagram for K42 Method.

of the A.B form and its acceptance on the feasibility test. The last S1 before R32 quits is a preliminary test for desirability, which in this case is vacuous and automatically S1. At this point, R32 is prepared to put together a subgoal to apply this form of R1 to L1. This is again checked to see if it has been created earlier, and the answer being in the negative, a signal S40 (new subgoal) is set.

We are again back in R11, which reads the S40 and evaluates the subgoal to see if it wants to do it. Again the result of the evaluation is S8; however, this time it stems from the fact that only reduce goals can be evaluated, since they are the only ones that have differences. Hence, all transform and apply goals are automatically evaluated S8, which is interpreted to mean "try it."

R11 executes Q81 and then 1Y90 which leads to the second recursion of R10, this time on Goal 3. Repeating the cycle of three S50's we are again in R11 executing the first segment of the method K41 for trying to apply an operator. A flow diagram is shown in Fig. 3. The method for applying operators is somewhat more complicated than the other methods for two reasons. First, operators are of various types——some are forms, some are IPL programs, some have side conditions, and so on. Hence the first step is to discriminate which kind of operator is being applied; the S61 indicates we are working with a form operator. (The alternatives have been left out of Fig. 3.) Second, operators can have more than one input. This leads to a host of complications, which show up in Fig. 3 as the production of modified operators rather than modified objects. Since no multiple input operators are used in this simple problem, we will ignore these various alternatives; however, it seemed necessary to put them into the figure.

Apply R to A

Method K41

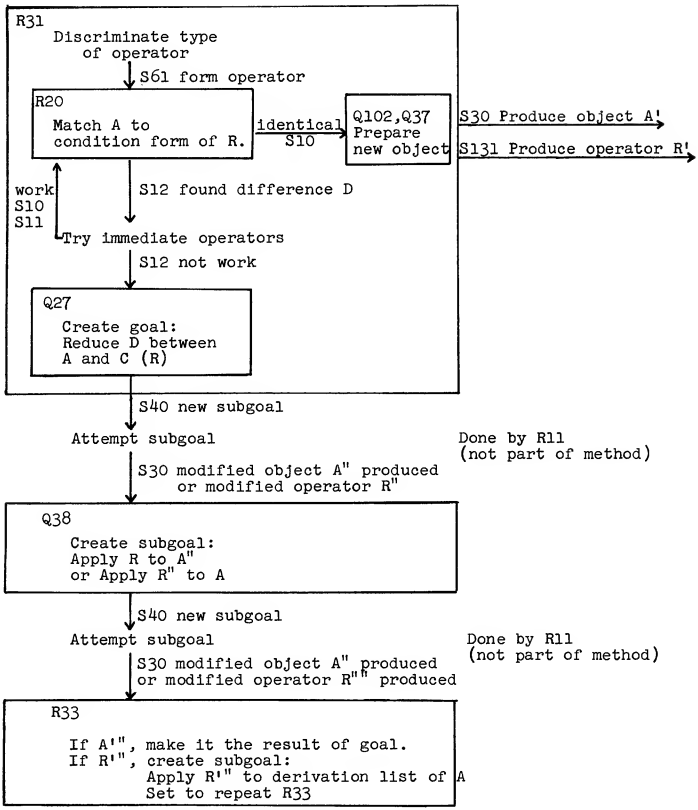


Fig. 3 Rough Flow Diagram for K41 Method.

A form operator is applied by matching the input expression against the condition form of the operator; i.e., R.(-PIQ) against A.B. This not only verifies that the conditions of the operator are satisfied (that the connective is a dot), but also gathers the information necessary to produce the new expression; i.e., A is R and B is -PIQ. If the match (R20) is followed through on the signal line, it will be seen that difference D15 is found twice. D15 stands for a variable versus an expression; it is one of the differences for which GPS has an immediate operator. Consequently, after R20 sets S12, R31 finds the substitution operator, performs it, gets the S12 changed to S10 (i.e., after substitution this part of the expression must be identical), and returns to the match routine. At the end, after the two substitutions, the condition form and the input expression are identical (S10) and so R31 gets Q37 to produce the new expression from the output form (here, B.A) which has now been filled in. Thus L2 is produced, and Goal 3 has been attained.

Before L2 was printed out as the result of Goal 3, a check was made to see if the expression, (-PIQ).R, had already been derived. This was done by checking each of the expressions on the derivation list (#28). In this case there was only L1, and so L2 was a new expression and L2 was added to the list.

At this point in the signal line, we have S30, indicating that Goal 3 was attained. This is detected by R11, serving as a signal for it to quit, and by R10, serving as a signal for it to quit. GPS then returns to Goal 2, as indicated in the signal line, and is back in R11. The 'Goal 2' is actually printed by Q82 in R11, which changes the goal context. R11 detects the S30 and sets S48, which is the sign that the subgoal in the method succeeded and that the next segment is to be obtained. (This takes an additional turn around the main R11 loop,

since it must be determined whether to go on to the next method segment [S41, which occurs here], or to repeat the previous segment [S46].)

From the flow diagram for method K42, we see that the next segment is just the trivial step of assigning L2 to be the result of Goal 2. Thus Goal 2 has been attained, and again S30 (success) is detected by both R11 and R10, so that GPS returns to the context of Goal 1. Again R11 goes through the motions of detecting the S30, setting S48, and finding that it wants to go to the next segment of method K40. This segment is Q28, which creates the goal of going from L2 to L0; i.e., the rest of the way after L2 (presumably) has taken the first step of eliminating the difference in position.

At this stage, we are back to familiar ground. The pattern of behavior for Goal 4 is identical to that for Goal 1 originally. A match is performed, which discovers a difference in connective between the left subexpression of L2 and that of L0. (The P9 in the goal expression indicates "lower left.") If the match is traced through, it will be seen that the comparison at the top level fails to find a difference (S11 following the first S20 in R20), so that the two left subexpressions were put into correspondence and the comparison routine (Q20) executed for them. Having found a difference, Goal 5 is set up to reduce this difference. At this point, the goal evaluation accomplished in R11 (at Y92) is effective. A change of connective on the left subexpression (D5 on P9) is compared with a change of position on the main expression (D9), with the conclusion that the former difference is smaller than the latter. This is reflected in the S7 following the S40 just before GPS attempt Goal 5. Until this time, there was nothing against which an evaluation could be made.

As in the earlier sequence, Goal 5 leads to a search for a relevant operator. R6 (AIB => -AVB) is selected because: 1) it is on the table of connections as changing connectives; 2) it has not been used before; and 3) it has the same main connective as the left of L2 (which is where it is to be applied). Again there is no difficulty in matching the condition form of R6 to the left of L2 and so L3 is produced.

The entire cycle repeats itself once more: obtaining L3 implies success on Goal 6, which in turn implies success on Goal 5. This leads to Goal 7 to transform L3 into L0, analogously to the creation of Goal 4. Attempting Goal 7 reveals yet one more difference, a change of position on the left subexpression, which generates Goal 8 to reduce it. Again the evaluation is favorable (S6) and Goal 8 attempted, leading to R1 (this time the AVB => BVA variant) and Goal 9. R1 can be applied, giving L4, which is the result for both Goal 9 and Goal 8. Finally Goal 10 is created, to transform L4 into L0. At this point, the match finds no more differences between them and so Goal 10 is attained (S30). This success rapidly propagates back up the goal hierarchy to Goal 7, then Goal 4, and then Goal 1. At this point, GPS realizes it has solved the problem and quits.

It should be apparent that there is a large number of features and responses of GPS that have not been illustrated. The most apparent example is that the operators always worked right away. Often, of course, when an input expression is matched to a condition form, a difference more serious than D15 (variable versus expression) shows up. The flow diagram for method K41 shows that GPS will then set up the reduce subgoal to try to eliminate this difference. In addition to this, all the goal evaluations were favorable, so that we never saw a goal rejection; likewise, none of the created goals and expressions

duplicated any structures already on hand. And as we commented earlier, no multiple line rules were applied. All these features, and a number of others, add variety, and sometimes zest, to GPS's behavior.

III. DATA STRUCTURES

There are several major kinds of data structures on which the program operates. For each a description of the structure, the conventions that govern its use, and a discussion of the ways in which it is created, modified, and destroyed is given. Various minor data structures, such as the reference trees, are defined and discussed where they naturally arise in the use of the major structures.

CONTENT TYPE

Some of the major kinds of data structure are labeled by a content type at A51. The ones currently defined are:

K161	Object TEX
K162	Operator TEX
K163	Set of TEX's
K170	Constants
K172	Primitive operations
K173	Variables
K179	Object types

GOALS

A goal is a collection of information that defines a desired state of affairs plus the means to attain this state of affairs and the history of previous attempts. All the information about a goal is on its description list; the list named by the goal symbol is always empty. Thus all information is obtained via attributes, usually G-symbols, but occasionally A-symbols, where the attributes are common across goals and expressions. The A-attributes used with goals are A2 (external name, an integer, which is the order of generation), A7 and A8 (used with auxiliary storage), and A18 and A19 (used in output). The

attributes are routines and are executed to find the attribute values. For the inverse operation of putting values on goals, three routines are defined:

Q13 Put (1) to be non-local value of attribute (0) of goal in Y2.

Q14 Put (1) to be local value of attribute (0) of goal in Y2.

Q15 Add (1) to front of value list of attribute (0) of goal in Y2.

Goal Types

Goals are of several types. Each type dictates the kind of information required to specify the state of affairs desired. Externally, a goal is specified by a simple list giving its type and the objects involved. This list is converted to a description list internally and all the additional information added to it (by E22). The current goal types, denoting the attributes and values used internally by A V without a separating comma are:

External		Internal
#1/	0	Transform expression B50 into expression B51.
	K1	G21 K1.
	B50	G1 B50, G11 P8.
	B51.0	G2 B51, G12 P8.
#2/	0	Apply operator B1 to expression B50.
	K2	G21 K2.
	B1	G5 B1.
	B50.0	G1 B50, G11 P8.
#3/	0	Reduce difference D1 between expressions B50 and B51.
	K3	G21 K3.
	D1	G4 D1.
	B50	G1 B50, G11 P8.
	B51.0	G2 B51, G12 P8.

The attributes G11 and G12 are for location programs, which locate the subpart of the expression that is being designated. Externally, subparts of the expression can be designated by putting a * next to the subexpression.

Goal Sufficiency

An important property of a goal is the sufficiency of its information: given an arbitrarily selected goal at any point in the course of problem-solving, it is possible without additional information, to commence problem-solving activity on that goal and to integrate the results of such activity with the rest of the total problem-solving activity. This means that it is possible to find out from a goal the kind of situation that is desired (G1, G2, G3, G4, G5, G11, G12, G13, G14, G15, G21, G31); the current state of solution (G20, G25, G30, G36, G39, G52, G53, G54); its role with respect to its supergoal (G23, G28, G29, G37); the kind of techniques available for attaining it (G27); its subgoals (G24, G25); and its relation to various other goals (G22, G33, G35, G38, G40, G50).

Goal Repeatability

A second important general property that goals have is their repeatability. A goal may be attempted any number of times; i.e., an attempt made to attain it. Each attempt by a problem-solving executive (currently R10) takes into account the previous history of attempts with the goal, and tries something different. If the goal has been solved previously, then additional attempts result (if successful) in alternative ways to attain the goal. For example, if the goal was to transform expression B50 into expression B51, then successive successful attempts would provide different ways in which this could be done; i.e., alternative proofs. It is possible, of course, that the opportunities for attaining a goal may be exhausted, either because all solutions have been generated or because more variations on methods and techniques would yield nothing new. In this case every

attempt to obtain the goal will yield a signal that indicates this state of affairs (such as S35 of S52).

Goal Context

The current goal is given in Y2. All information in the Y-cells is relative to this goal. Thus, several other Y-cells contain goals:

Y7	supergoal (K90 if not exists).
Y9	most recently tried subgoal (K90 if not exists).
Y10	equal goal (may not exist).
Y87	proposed goal (held here until determine if should be next in Y2).
Y88	temporary cell for prior goal (needed while establishing new goal).
Y111	top goal (this is not relative to current goal).

Goal contexts are changed by one of a set of routines, Q81-Q87. Each of these establishes a goal under certain conditions: setting up a new subgoal (Q81); setting up arbitrary goal for a retry (Q83, Q85, Q86, Q87); or returning to the goal from which the current goal was tried (Q82, Q84). All the goal setting routines use a common subroutine, Q80. This routine sets Y2, Y3, Y4, Y7, Y9, Y34, and Y86. In addition these routines establish the method-segment context,* in which a goal was operating (Y5, Y6) where this is required (Q82, Q85, Q87); adjust the relative depth (Y35); and set the signal (Y1) to be the goal status (G20). These eleven Y-cells, plus those that are goal invariant by definition, are all the Y-cells that can be relied upon to hold good information at the beginning of an attempt on a goal.

*See discussions on methods, Sec. IV.

Goal Creation and Destruction

Goals are created by various Q-routines (Q27, Q28, Q34, Q38, Q40, Q103, Q108, and R1, the latter being a temporary expedient). Each goal creation starts by giving the type of goal desired (K1, K2, K3) as input to Q16. Q16 obtains from the goal type a form for that goal (A20 with values K11, K12, and K13 respectively). These forms are copied (J74) to produce the basic information for a new goal and then Q16 links the new goal to its supergoal and records the method-segment context in which the subgoal was created. The specific goal creating routine records the particular components (G1, G2, etc.) used to define the goal. The final step in goal creation is Q17, which records on the various components information about the goals with which they are used.

Goals are independent structures. Whenever a goal name occurs on another list, such as the name of a subgoal on the G24 list of its supergoal, it is always nonlocal. Thus, if a goal were to be erased (J72), no other goals would automatically be erased as a consequence (although access to them might be lost). Currently, goals are never erased once created. Instead they are filed on auxiliary storage when space becomes scarce.

Goal Identity Test

An important step in creating a goal is to determine if this goal already exists. Q17 makes this check, using Q46. There is a goal reference tree (in Y25) in which all goals are recorded (by Q46). This is a branching structure, corresponding to a variable pocket sort, which is built up by Q46 as the set of goals increases. The goals are first sorted by type (G21), then they are sorted on the name of the first expression (G1). All those goals with the same G21 and G1 are put on a simple list.

Thus we get a structure:

```
Y25/9-0.0   9-0/9-1.0      9-10/9-11.0      9-100/0
            9-1/0          9-11/0                C32
                 K1             B50                7010.0
                 9-10           9-100
                 K2             8425        9-110/0
                 9-20           9-110.0           11320.0
                 K3
                 9-30.0    9-20/9-21.0      etc.,
                           9-21/0
                                B50
                                9-200.0

                           9-30/9-31.0      etc.,
                           9-31/0
                                B50
                                9-300
                                12345
                                9-310
                                8425
                                9-320.0
```

Q46 takes a new goal and locates it in this tree structure. If there are any other goals in the same pocket it tests the new goal against each one on the remaining attributes needed to determine if the goals are the same (G11, G2, G12, G4, G5). If the goals are not the same, it adds the new goal to the list and reports back S40 (new goal generated). If no competitors exist, of course, the goal is established as the first member of its (new) pocket.

Goal Duplication vs. Equivalence

If the new goal is identical to some already existing goal, then there are two cases: either the goal is essentially a duplicate and GPS does not want to attempt it (indeed, it wants to clip the goal tree at this point); or the goal, although identical, has been generated in a different context for a different purpose. In this latter case, GPS can attempt the new goal with profit and should use any results that might have already been obtained on

the existing goal. Routine Q71 distinguishes between these two cases.

The current criteria of duplication (leading to S54) are: goals that are of type K1, or have the same super-goal, or have the same super-supergoal unless goals are of type K3. All other cases are taken to imply useful or "equivalent" goals (and lead to S42). In this latter case Q71 sets up a way for the two goals to borrow results back and forth. For each set of equivalent goals (there may be more than two) there is a list. This is on each member goal at G38. This list has on its description list at A14, a list of all the results obtained so far by all the member goals. These results are two-item lists: the G3 component, followed by the G13 component. There is a method called the Transfer Equivalent Result Method (K43), which Q71 establishes as part of the method list (G27) of each member goal. The section on method K43 should be consulted for the details; Q71 simply sets up for this method at the point when a new goal is found to belong to an equivalence list.

Goal Modification

Goals are modified by innumerable routines (the inverse listing for Q13, Q14, Q15 indicates the occasions). They are never destroyed once they are created, but can be stored out on auxiliary storage and only called in when they are needed again. (See Sec. V on auxiliary storage.)

EXPRESSIONS AND OBJECTS

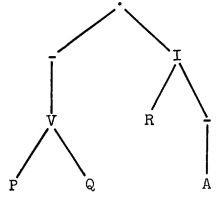
The objects that GPS manipulates and the operators with which it manipulates them are specified by expressions. All expressions, regardless of the particular TE in which they occur, satisfy a certain set of conventions as to how they are encoded into list structures.

Structures of TEX's and EX's

The TE is conceived to consist of many, independent objects. The expressions which describe these objects are called Total Expressions (TEX's). Each expression may be built up from many subexpressions; each of these is called simply an expression (EX). An expression consists of a hierarchical structure (a tree) of subexpressions related together at each level by some operation or connective. By conventions each node of the tree is given by a simple list in which the head contains the operation or connective and the list cells contain the names of the subexpressions. Thus in symbolic logic we would have:

external form: -(PVQ).(RI-A)

tree form:



list form:

9-1/.
9-2
9-3.0

9-2/-
9-20.0
9-3/I
R
9-30.0

9-20/V
P
Q.0

9-30/-
A.0

(Since the computer alphabet is limited, the I stands for "implies," usually denoted by $\supset$ or $\Rightarrow$.) Notice that none of the expressions is describable and that they all form a single list structure (i.e., their names all occur as local symbols). On the other hand, a total expression

(TEX) is describable. Its description list contains information about the expression, its history, genesis, and properties. The actual expression (EX) that the TEX represents is given in the first list cell and is called the main expression. Thus, in the example above, if -(PVQ). (RI-A) was to be a total expression, called 7155 say, it would appear as:

```
7155/ 9-0
      9-1.0                    (9-1 is the same list as above)
 9-0/ 0
      A2
      9-100=  3                7155 was the third expression
      A3                                created
      9555                     7155 was created by goal 9555
      A4
      9-110   9-110/0          7155 used by goals C32 and 702
      A12           C32
      K70           702.0      7155 belongs to TE K70 (symbolic
      A13                               logic)
      9-120   9-120/0          7155 has one variable, A
      A15           A.0
      9-130=  9                Complexity of 7155 is 9 (number
      A16                               of nodes)
      9-140=  4                Maximum depth of the tree is 4
      A51
      K161.0                   Object TEX's are of type K161.
```

The necessity for the distinction between main expression and total expression arises because we make description lists obligatory on TEX's but do not permit them on EX's, storing the operation symbol in the head of EX's instead.

The external format for TEX's is a simple list:

```
B88/0
    -
    (
    P
    V
    Q
    )
    .
    (
    R
    I
    -
    A
    ).0
```

There is no way to input an EX, since it cannot exist by itself.

Creation and Destruction of TEX's

TEX's are created by initial input from outside or by the application of operators to TEX's that already exist. Once created, a TEX is never modified and never destroyed. Creation is always done by P50. It involves an "official act" of assigning a name (A2, which has the order of generation as value), and recording certain information about the TEX (A13, A15, and A16 currently). Expressions may exist temporarily and then be erased, if they are no longer wanted. However, these are not TEX's. Only when an expression is put on a goal (a G1, G2, G3, or G5 currently) does it become a TEX with a name and, hence, unmodifiable. A check exists in the system in Q17: all components being put on the goal must "exist"; if not, they are given permanent status at that time by Q17.

As discussed in additional detail in the section on Matching, the Y-cells used to hold expressions while they are being worked on are Y11-Y13-Y15 (for the first expression under consideration) and Y12-Y14-Y16 (for the second expression under consideration). In the first case, Y11 holds the location of the expression; Y13 holds

the TEX (that is, the name of the independent entity containing the subpart in Y11); and Y15 holds the location program that locates the part of the TEX initially considered. A similar interpretation holds for the second expression. When setting up a process, such as the matching of the expression at G1 to the expression at G2 during an attempt on a K1 (transform) goal, the G1 TEX is put into cells Y11-Y13-Y15 and a J3 is put into Y45. This latter symbol indicates that the expression in cells Y11-Y13-Y15 is an official TEX and cannot be modified. It can be examined by the match process without restraint, but if ever a modification occurs, a copy of the expression in Y11-Y13-Y15 is generated, replacing the G1 TEX, and this copy is modified. Concurrently, a J4 is put into Y45, which indicates that the expression in Y11-Y13-Y15 is no longer an "existing" entity. Consequently, subsequent modifications can be made in the expression in Y11-Y13-Y15 without additional copies. Only when some routine (Q27, Q34, Q37, Q40) uses the expression on a goal is it made into an official TEX and a J3 put back into Y45. An entirely similar situation exists for Y12-Y14-Y16 using the cell Y46. At the completion of any processing of expressions in the Y-cells, a clean-up routine (Q24) is executed. This erases any expressions named in the Y-cells where Y45 and Y46 indicate that it does not have official status. The routines Q11 and Q12 are the ones that check Y45 and Y46, respectively, copy (P13), and replace the expressions in the Y-cells if they need it. Q11 or Q12, as appropriate, is executed at each point where it becomes certain that an expression will be modified (Q11 in F4, F28, Q37, and Q52; Q12 in F5, F29, Q37, and Q51).

Some other expressions (the operator at Y20 and the difference expression at Y84) also use an indicator (Y47 and Y48, respectively) to indicate whether they have "official" existence or are to be erased by Q24.

DERIVATION LISTS

There is no reference tree of objects, analogous to the reference tree of goals, even though each object must be checked to see if it has already been created. This role is played by derivation lists. Each expression is generated from some other expression (or expressions) by means of an operator. In general all those objects deriving from common parents are interchangeable in their role as starting points in the application of additional operators. Hence, as each expression is created it is put on a single list, called the derivation list. For all expressions with a common parenthood this same list can be obtained at A5. It is possible for several derivation lists to exist, however; one working forward from the givens, one working backward from the desired, one holding operators that have been derived from other operators, one starting from a conjecture that was tied neither to the given nor the desired, and so on. Derivation lists are created (P58) every time a TEX is created that is unrelated to any of the derivation lists already in existence. At the beginning, this is done in Q45.

The derivation list is a TEX:

```
11245/9-1
     9-2.0  9-2/,          , is the connective for
                B50                    "set"
                7155
                8266.0

  9-1/0
      A51
      K163                  Derivation lists are of
      etc.                        type K163
```

When a new expression is created, it is checked for identity against all the expressions on its derivation list (Q43). If it is really new, it is put on the list

and S30 reported. If it already exists, the new version is destroyed and S36 is reported along with the name of the old version.

OPERATORS

The operators are also TEX's, but may be of several kinds as indicated by A1 which takes on signals as values.

Form Operators

S60 indicates a form operator with some initial condition that has to be tested by a program (at A10). After this test is completed (see R31), the operator can still be any of the several kinds. S61 indicates a form operator, without such conditions, such as exists in logic. The left-hand subexpression of a form is the condition form which must be matched to the input object. The right-hand subexpression is the product form which gives the expression that is to replace the input expression. Unless otherwise stated, a form operator may be applied to any subexpression (EX) of a TEX. For example, B20 is the operator A.B Y B in logic. (Since the computer alphabet is limited, the Y stands for "yields," usually symbolized by => .)

```
B20/9-1
    9-2.0      9-2/Y
                   9-3    9-3/.
                   B.0          A
                                B.0

9-1/0
    A1
    S60
    A10
    F2
    A51
    K162
    etc.                          Operator TEX's are of
                                        type K162
```

A1 S60 indicates that a test must be performed, in this case F2. F2 is an IPL-V routine, which if the expression being operated on were a main expression and positive, would result in the signal being set S61, which would then indicate that B20 is a form operator.

Expressions for Operators

It is also possible for an operator to be given an expression, such as, "the reverse operator to B12" (S62). Such an expression is itself a TEX:

```
B13/9-1
    9-2.0      9-2/K60          K60 is the operation,
                  B12.0                  "reverse"

9-1/0
    A1
    S62
    etc.
```

Before such an operator can be applied, it must be expressed more directly. K60 has associated with it a program (P30 at A11) that will be the operand (B12 here) and create a new operator that is the reverse of it; i.e., has B12's product form as condition and B12's condition form as product.

Direct Operators

Finally, it is possible to have a direct operator (S63), which is given simply as an IPL-V program (at A11). Such operators may have additional input information given as an expression, but it is the routine at A11 that manipulates this information, not the general purpose GPS routines for manipulating forms. Missionaries and Cannibals provides an example. A TEX in M&C looks like (where B = the boat, M = a missionary, C = a cannibal, L = the left side, and R = the right side of the river):

```
M72/9-1                                              M&C  TEX
    9-2.0     9-2/+
                  9-3   9-3/L   9-4/R      Left side : MMCCB
                  9-4.0   M       M        Right side: MC
                          M       C.0
                          C
                          C
                          B.0
```

An operator for Missionaries and Cannibals looks like:

```
M30/9-1                                    Move MC from the left
    9-2.0    9-2/Y                         to the right side (a
                 9-3.0   9-3/L             direct operator)
9-1/0                      M
   A1                      C.0
   S63
   A11
   M22                                     M22 is general operator
   etc.                                                routine
```

The operator specifies a general routine, M22, at A11 and provides an input form to tell M22 what specific action to take. M22 interprets this form to mean, "test if the boat is on the left side; if it is, take one M and one C from the left-side list of the TEX and move them to the right-side list of the TEX." In the case of M72 this could be accomplished; in other cases one of the symbols (B, C, or M) might be missing and M22 would terminate with a difference.

LOCATION PROGRAMS

The TEX is the independent unit, consisting of a hierarchy of subparts. GPS is concerned with the various subparts (e.g., it can apply operators to them) and requires a way of designating them. It needs a way that is independent of the particular names used for a subpart (i.e., of the addresses); e.g., it must find corresponding places in two expressions (such as a TEX and its copy). In addition, it must be able to store the TEX out in auxiliary storage and still find the same subpart after

retrieval. (Only the name of the TEX is preserved when a structure is filed on auxiliary.) The device used is called a location program. It is an IPL-V program which, if applied to the TEX in HO, gives (in HO) the location of the subexpression designated.

Inputs Are Locations Not Names of EX's

Before discussing in detail the structure of location programs, it is necessary to observe a major convention about routines that work with expressions: the appropriate input and output to such routines are the locations of the expressions to be worked on, rather than the expressions themselves. Thus, for example, Y11 holds the location of the expression designated; P15 generates the locations of all the terms in the expression whose location is (1); P26 tests if the expressions located in (0) and (1) have the same terms; and so on.

This convention is necessary to permit the modification of expressions. If only the name of an expression is available in HO, it is not possible to change the occurrence of the name in the higher expression. That is, the list cell in the higher expression that holds the name is no longer accessible.

Structure of Location Programs

The location program is a simple list composed from two routines:

P8 Locate the next EX after the EX located by (0);

P9 Locate the first EX in the subexpression of the EX located by (0).

P8 is just a J60 and P9 is just a J80 followed by a J60. Any position in a hierarchical list structure can be found by executing a sequence of P9's and P8's. A

location program is always executed on the name of a TEX in H0. Thus in the expression, 7155/ -(PVQ).(RI-A), we get the following location programs for locating each subexpression (all with input 7155):

```
                                        7155/9-0
P8                                          9-1.0
                                         9-1/.
P8  P9                                      9-2
P8  P9  P8                                  9-3.0

                                         9-2/
P8  P9  P9                                  9-20.0

                                         9-20/V
P8  P9  P9  P9                              P
P8  P9  P9  P9  P8                          Q.0

                                        9-3/  I
P8  P9  P8  P9                              R
P8  P9  P8  P9  P8                          9-30.0

                                        9-30/ -
P8  P9  P8  P9  P8  P9                      S.0
```

Notice that P8 is the location program to be applied to the name of the TEX (not its location) to get the location of the main EX.

Location Program Reference Tree - Absolute

The same location programs arise over and over again, and it is desirable to have fixed names for each location program that is used (thus permitting the identity test for location programs to be a J2 test on their names). This tree provides a node for each location program that exists. If location program X is applied to K98 (as a TEX), then it will locate the node corresponding to X. At that node can be found the name of the canonical location program (at A60) and also the number of levels down in the tree (at A61). Thus, whenever a location program is constructed (e.g., during a

match), it is executed on K98 to find the canonical location program, and that name is used. By the way P8 and P9 are constructed, if a location program is ever executed on K98 that does not correspond to a node existing in the tree, the reference tree will be automatically extended and the necessary canonical location programs will be created.

Some of the location programs occur as P-routines, since they are used in programming parts of GPS.

```
P10/P8      Locate first subexpression
    P9.0    (also, locate operator condition)

P11/P8      Locate second subexpression
    P9      (also, locate operator product)
    P8.0

P71/P8      P72/P8
    P9          P9
    P9.0        P9
                P8.0
```

Location Program Reference Tree - Relative

Besides having programs that take the TEX as input and deliver the location of the desired expression as output (called absolute location programs), it is also desirable to have location programs that take the location of an EX as input and locate some EX below it. These latter are called relative location programs. Due to the conventions for TEX and main EX, relative location programs differ slightly from absolute ones.

```
P9          Relative location program:  first subEX, one
                                            level down
P70/P9      Relative location program:  second subEX, one
    P8.0                                    level down
```

Consequently, the relative location programs have their own reference tree (K99).

There are a few routines for manipulating location programs (P40, P46, P47).

DIFFERENCES

The differences are symbols which are associated on the one hand with the difference between expressions, as discovered by the match routines (R20, using Q20 and Q47); and on the other hand with the operators (via the table of connections in Y52). At the moment, they have no information at all associated with them.

IV. ROUTINE STRUCTURES

The performance of GPS after it has been given a problem by the experimenter can be described by starting at the executive routine, which is at the top of a hierarchy of routines, and working down through successive levels of detail.

TOP EXECUTIVE

The top routine of the experimenter (E2 currently) sets up the initial goal in Y87 and executes the top executive of GPS-core (R2, normally, or R1). R2 sets up the goal context, creates the equivalence lists (Q45), initializes the various limits (Q44, Q107), and records the initial goal on the goal reference tree (Q46). It then executes the problem-solving executive in cell Y90 (R10). The top executive, as a temporary expedient, uses the Y's directly (in violation of the conventions for R-routines).

PROBLEM-SOLVING EXECUTIVE

The important executive is R10, which is used recursively in attempting each goal. A flow diagram for R10 is shown in Fig. 4. The flow diagrams for the R-routines depend on the convention that each step consists of executing a Q-routine and then taking a multiway branch on the basis of the signal. Indirect executions (e.g., 1Y90 and 1Y96) occur in several places to make it easy to change key routines. These Y-cells are occupied either by Q- or R-routines.

Centralization of Decision-Making

The signal system provides the basis for the most important convention about the way GPS operates: all

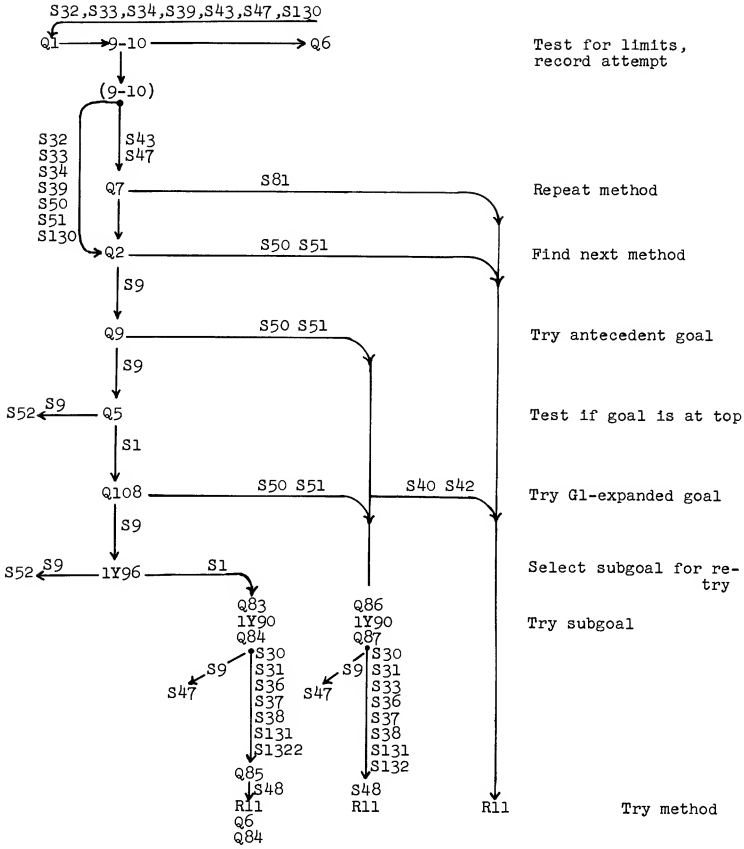


Fig. 4 R10: Problem-Solving Executive.

important decisions are made by the problem-solving executive, rather than being delegated to lower routines. Thus, no matter where in the routine hierarchy a crucial decision is posed (e.g., whether to attempt a subgoal), it is necessary to bring this decision back to the executive. The signal system can be viewed as providing a symbolization of all the important decision situations that occur in the course of operation. A Q-routine represents a simple enough action (in terms of the decisions which must be made to carry it out) that GPS can commit itself to carrying through a Q-routine once it has initiated it. Thus Q-routines represent, in a way, the "unit actions" of GPS. When a Q-routine is executed, the "unit action" is carried out, and information about just what happened is reported back as a signal (by an S-symbol), so that a decision can be made about what to do next.

Control Techniques to Handle Centralization

Actually, the important decisions are shared among all the R-routines, and are not all localized in R10. However, the tendency remains for decisions to be "kicked upstairs" for solution. This implies a certain violation of the hierarchical organization of processing, since it often happens that a crucial decision (such as whether it is worthwhile to continue) occurs in the middle of a process. It is then necessary to leave the routine to return to the higher routine for decision and then (if the decision is to continue) to return to the lower routine again. Mechanically this is accomplished in GPS by one of two mechanisms. The process may be split into several Q-routines, so that the subroutine hierarchy is formally preserved. This results in Q-routines with rather truncated functions; i.e., just a fragment of what

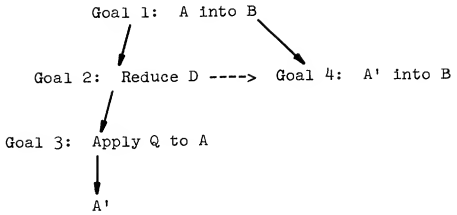
would normally be incorporated in a subroutine. The second technique is to begin R-routines with discrimination on the signal. Then they can be entered several times with different signals, which then cause an immediate transfer to the appropriate starting or continuing place (see R11 and R20). No difficulty arises in all of this, but it makes the operation rather confusing until these features of the program are understood.

Structure of R10

As revealed in Fig. 4, the operation of R10 is rather easily comprehended, reflecting the crudity of the ideas about how to handle the top decisions. R10 consists of a loop: Q1 — select an attempt — attempt it — Q6 and recycle. Q1 tests whether any "external" limits have been transgressed, such as effort limits (S72) or depth limits (S74). It also sets up a record of the attempt (Q72). This is a structure which is filled in by Q6 at the end of the attempt and recorded on the goal at G26. The vertical column at the right side of Fig. 4 represents the choice of what attempt to make. Right after Q1, it is possible to attempt to repeat the methods just tried (Q7); to select another method from the method list (Q2 using Y5, which is obtained from G27); or to quit any further attempts on this goal (all S-symbols not occurring at the branch point).

Antecedent Goal

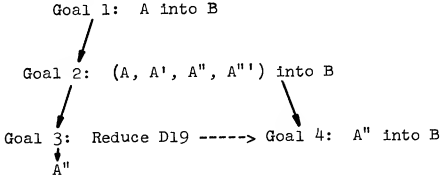
As the next alternative, the antecedent goal to the most recent subgoal may be retried (Q9). An antecedent goal is one that produced a result used in defining a goal. Thus, to retry it is to attempt to get an alternative result; if successful it will produce a modified goal which may be more tractable. In a typical goal tree:



Here Goal 2 is antecedent to Goal 4. Retrying it will lead to some different operator, Q', being tried; hence (perhaps) to some different expression, A", being produced; and then to a new goal, A" into B, being formed which is an alternative to Goal 4.

G1 - Expanded Goal

Two other alternatives for attempting a goal exist in R10. Both of these are currently restricted to the top goal (Q5 exits S2 if not top goal). One possibility is to generalize the goal by using the derivation list of the G1 expression in place of the expression. For example, if the top goal were to transform A into B and A had expressions A', A", A"' on its derivation list, then the situation would develop as follows (Q108 plus subsequent problem-solving):



Setting up Goal 2 to transform the set into the expression generates a D19 difference (set versus element); this in turn causes the selection of one of the elements of the set (Q54), in this case A"; and this in turn leads to creating the succeeding goal, A" into B. This technique of generalizing a goal is a way of seeing if some of the expressions which may be substituted for A might not be better starting points than A for reaching B. Some of these other expressions may have been generated without their relationship to B being considered (e.g., as a modification of A in order to apply an operator). Thus, it is sometimes possible to discover the possibility of a useful expression among the ones already generated. Although we have stated this notion with respect to transform goals, the same concepts apply to the other two goal types. This technique is inserted directly in the executive as Q108, rather than as a method. This is an expedient to restrict its usage to the top goal.

Lower Goal Selection

The final possibility for attempting a goal is to select a lower goal and try it again (1Y96, which normally holds Q109; but also Q8). A lower goal is one that lies anywhere in the goal tree headed by the given goal; or, alternatively, a direct or indirect subgoal of the given goal. The goal selection procedure involves first listing all the goals that are either untried or unfinished (G20 S50 or G20 S51 in Q109). This list is then submitted to a further selection (Q73 or Q105). Q73 selects the best subgoal according to the goal values (G22) and the goal evaluation procedure. Q105 first splits the goals into those which are subgoals of transform goals and those which are not. It tries to get a goal from the former sublist first (with Q73). Only if no evaluable goal exists

on the K1-subgoal-list, does Q105 select from the remainder list.

Once a goal has been selected it is retried, independent of the reasons for its not being tried further at the time it was last worked on. Thus, this procedure "forces" its way past the tests for rejection of a goal. The net result is to make the total problem-solving activity of GPS proceed as a series of episodes, each one starting from some goal that already exists in the goal net and trying to extend it further until the various limits and rejection criteria force a halt to the exploration. After each episode GPS reselects another goal somewhere in the total net to start the next episode.

Execution of Selected Attempt

R10 does not choose among all these alternative ways of attempting a goal in a very sophisticated way: the vertical column in Fig. 4 implies an approximately lexicographic ordering. Once an affirmative decision has been made about an attempt, then it is carried out by R11 or by attempting a subgoal, as appropriate. The subgoal-attempts require using a Q8x routine to get into the context of the subgoal (Q83 or Q86), executing the executive (1Y90, holding R10), then returning to the context of the present goal (Q84 or Q87).

In both cases of subgoal-attempts, R10 is retrying a goal that was created on another occasion, tried (perhaps), and abandoned. Thus there are now three goals involved: the current goal, the lower goal it wants to attempt, and the supergoal to the lower goal, which created it and, alone, has the information to utilize its attainment. If the lower goal is an immediate subgoal (as in the antecedent goal), then the current goal and the supergoal coincide; otherwise they differ. If the

attempt on the lower goal fails, then in all cases we wish to return to the context of the current goal. But if the attempt on the lower goal succeeds, it is necessary to get into the context of its immediate supergoal and to continue problem-solving from the position in that supergoal's activity which is prepared to use the results. This supergoal, of course, is still a subgoal of the present goal. This is accomplished by Q85-R11-Q84. Notice that there is a return afterwards to the context of the initiating goal, since it is in this context that the decision must be made as to whether to continue or not. If this supergoal (of the initially attempted lower goal) is able to make some progress using the result, getting yet another new result, it is then necessary to get into the context of its supergoal to use the just obtained result. Thus, there is a loop around Q85-R11-Q84, which continues until some supergoal up the line fails to make positive use of the results and the whole attempt comes to an end. Alternatively, of course, all goals are successful until the supergoal being put into context by Q85 is the present goal, which initiated the whole attempt series in the first place (S1 at Q85); in this case it is appropriate to remain in the present context.

In the case of Q9 and Q108, which generate immediate subgoals for retrying, it is not necessary to use Q85 to obtain the supergoal, since it is known that the current goal is the supergoal. Hence R11 is executed directly after a success on the attempted subgoal.

In the case of Q7, Q2, and sometimes Q108, we are either trying a method or working with an untried subgoal. Hence we go directly into R11.

Recording Attempts

No matter where we quit in making the attempt in R10, we return to do a Q6, which records the results of the attempt. There is then opportunity to recycle and continue with another attempt on this goal, or to quit this goal and go back to the supergoal. Q6 does not change the signal, so this decision is made on the basis of the final signal resulting from the attempt.

Q6 records in the attempt record (created by Q72) the signal that terminated the attempt (A40, the attempt status), the method used (A41), its current status (A42; see section on methods), and the limits of the attempt if they started or ended somewhere in the middle (A43 and A44). The attempt record is stored away at the front of the history of attempts (G26). In addition Q6 updates the method status: it changes it from S50 to S51, or from S51 to S52 if the method quit with S35 (impossible).

METHOD EXECUTION AND R11

Methods and Method Status

The main course of GPS is guided by a series of methods (K40-K44). These are associated with goal types, each method being a way to either attain the goal of the given type or to analyze the task into subgoals and use their results in attaining the goal. With respect to a given goal, each method has a status indicating whether this method has not yet been used (S50); has been used but may be used again (S51); is no longer useful (S52); or is temporarily blocked from being used (S53). Each goal has a method list (G27) which contains the name of each method applicable to the goal followed by the current status for the goal. Status symbols are updated by Q6 after each attempt. The set of methods is not completely

fixed, although it is initially determined by the goal type (the initial G27 list is on the goal form, A20, that is copied to create a goal). For example, K43, the transfer result method, is added to the list by Q71.

Method Structure: Segments

In the published papers on GPS (and in GPS-1 program) the methods are given as subroutines. However, the requirement that all important decisions be reserved to the problem-solving executive implies that the method be broken up into a number of method-segments. Between each segment, control returns to the executive (R10 and R11). Thus methods are data lists of segments and R11 acts in many ways like a higher level interpreter, executing each segment on the list in turn and making the decision whether to continue or not after each segment is finished. So far it has been sufficient to have methods be simple lists of segments which are executed strictly sequentially with possible repetitions. It has not been necessary to have the methods be branching, conditional structures.

Method Interpretation: R11

Figure 5 gives the flow diagram for R11, the R-routine that interprets and carries out methods. It consists of a loop through a single large branch list which distinguishes numerous signals. Typically R11 is entered from R10 with the status of the methods as the signal. If this is S50 or S51 (untried or unfinished) then the first method-segment is obtained (Q3); if the status is S52 or S53 (finished or blocked) R11 quits without doing anything (S52 and S53 do not appear in the branch list). If R11 is entered with S81, this symbolizes the repetition of a method (from Q7 in R10) and again leads to executing the first segment.

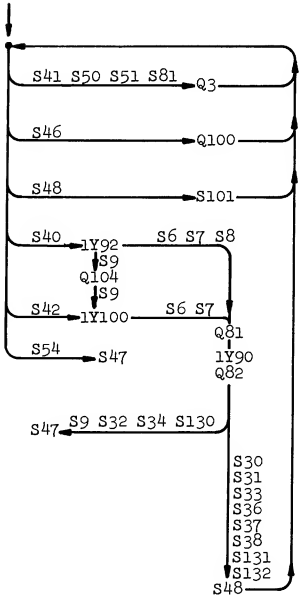


Fig. 5 R11: Executive Method until Fail.

Q3 obtains each successive segment of the method and automatically executes it (thus it performs the "fetch" and "execute" steps of a standard interpreter). The resulting signal is discriminated by the same major branch list. The results of a segment that R11 recognizes are to go on to the next segment (S41); to repeat the current segment (S46); to find out whether to repeat or go on (S48); that a new subgoal has been created (S40); that a new subgoal has been created which is equivalent to an existing goal (S42); and, that the method cannot possibly work (S35). Each of these leads to an appropriate routine, which result is again discriminated for further action. Several kinds of failing signals can occur (S32, S39), but R11 quits in these cases, so they do not show up in the branch list; the same is true of signals which indicate goal success (S30, S31, S37, S38, S44, S45, S131, S132).

In case a subgoal is generated (S40, S42) it is necessary to evaluate it before attempting it (1Y92 and 1Y100, normally holding Q74, but sometimes Q106, or Q4). This evaluation (see below) results in a signal (S3, S4, S5, S6, S7, S8), and for the appropriate set (S6, S7, S8) the subgoal is tried (Q81-1Y90-Q82). Afterwards, the results are summarized either as a failure (S47 and quit R11) or as a success (S48 and recycle in R11).

It is also possible to enter R11 with a subgoal to be tried (e.g., with S40 or S42). Then R11 starts with the evaluation and attempt (if appropriate) of the subgoal, followed by the other method-segments called for by Q3. The only unmentioned action is setting S47 if the subgoal is a duplicate (S54).

Goal Values and Goal Evaluation

In the published accounts of GPS, mention is made of "progress tests," which determine if a goal should really be tried. These are embodied in the goal evaluation routines (Q74, Q106, Q4). Certain goals can have a value (at G22); currently only reduce-type goals (K3) have values, the other (K1, K2) being unevaluable. Two goals with values can be compared (P56) with one of several results (S3, S4, S5, S6, S7); if either of the goals is unevaluable, the result of P56 is "undefined" (S8).

Goal Values

There are several kinds of values, each kind being identified at A83 by a structure (K101, K102, K103) which tells how to process the value. A K101 value compares first on the level at which the difference occurs (A84), giving S3 or S7 if the levels are unequal. If the levels are the same, it compares on the difference symbol (A89), giving S4, S5, or S6, depending on whether the first value is less than, equal to, or greater than the second value on the difference ordering (e.g., C10 for TE K70). A K102 value compares first on the number of levels up from the bottom of the expression (the maximum level minus the level) (A85) and then on the difference symbol (A89). A K103 value compares first on the difference symbol (A89) and then on the level (A84) (just the opposite from K101). Each of these values was introduced by experience with certain special situations; none of them seems to be appreciably better than the others. Each of these value types has on it (at A17) the appropriate comparison routines (P49, P57, and P59 respectively). Values of each type are created by separate routines (Q76, Q77, Q78 respectively); creation occurs in the Q-routines that create K3 goals (Q27 and Q40, by executing 1Y95).

Goal Evaluation

The evaluation of a goal to determine if it should be attempted consists first of a search for the goal against which the candidate should be compared, and second, of the comparison by P56 as described above. The search (in Q74) consists in finding the first evaluable antecedent goal, or supergoal. The rationale is that a subgoal should be less difficult than its supergoals, since it purports to solve only part of the total problem. Similarly, if GPS works from hard differences to easy differences, then a goal should be less difficult than its antecedent goals. Finally, of several potential antecedent and supergoals for comparison, Q74 prefers near supergoals to more distant (higher) supergoals, and antecedent goals to supergoals.

MATCHING

A crucial part of two methods (K40, K41) is the process of matching two expression together in order to determine their differences (or identity). There are two alternative match routines (R20, R21), of which R20 is the easier to understand and will be described first.

Matching is factored into two parts: two expressions are first put into correspondence; then the contents of various corresponding EX's are compared. Since all expressions have a common form (the tree structure with the operation in the head) a set of GPS-Core routines handle the task of putting two expressions in correspondence and cycling through the successive pairs of corresponding cells (Q21, Q22, Q23). For each pair a TE routine (in Y17) is executed which compares the expressions at that point (Q20).

R20 Match

The flow diagram for R20, shown in Fig. 6, consists of iteration through a basic discrimination, where Q20 is used to compare EX's (following S20, meaning "both found"). The other Q2x's are used to locate the next corresponding EX's in response to whether two cells are found to contain identical subexpressions (S10) so that further subexploration was unnecessary; whether no difference was found for this pair at this level, but exploration of the subexpressions should occur (S11); or whether a boundary of the expression has been reached (S23). Several negative signals are possible (S12, S13, S16, S21, S22) which do not appear in the discrimination since they imply that R20 should terminate. S19 is a signal indicating the beginning of a match (not, however, just entry into R20, since R20 may be executed and terminated numerous times during a single, successful match). At this occasion R20 executes a special comparison (Q47) of the top elements of the expressions being matched for differences directly recognizable by GPS-core (D19, D20, D21 currently). The final exit at S10 at the bottom of the diagram indicates an inference that if the expressions have been thoroughly scanned (Q23 yielding S23 indicates a return to the top of the expressions being matched) and no differences have shown up at any point (always S11), then the two expressions are identical (S10).

Housekeeping for Match

The match occurs by setting one expression to be matched (the "#1" or matching expression) in Y11-Y13-Y15 and the other ("#2" or the expression being matched to) in Y12-Y14-Y16. The initial setup (Q25, Q36) puts the TEX into Y13 (for #1); puts the location program that locates the EX into Y16; and uses this location

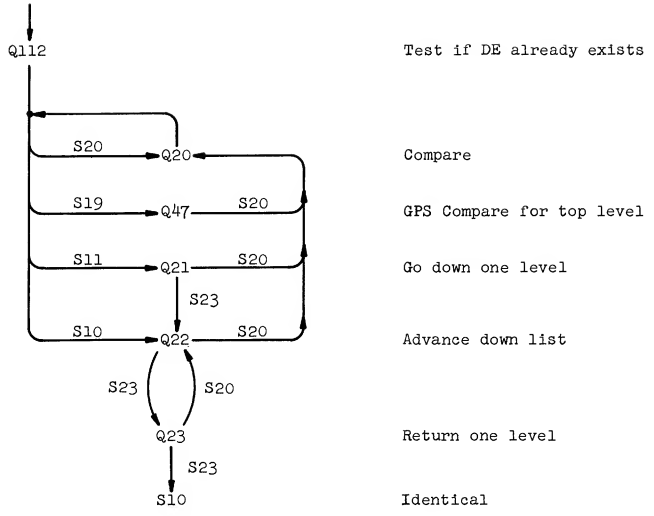


Fig. 6 R20: Match Element by Element, Depth First.

program to put the location of the EX into Y11. A similar setup occurs for the #2 expression. The location programs for the two expressions need not be the same. For example, in applying an operator, the main EX of an object (P8 location program) is matched against the condition form of the operator (P10 location program).

The movement through the tree structures of the two expressions (Q21, Q22, Q23) involves pushing down Y11 and Y12 as the scan goes deeper and popping up these cells to come back up a level (this is due to the one-way nature of lists). It is also necessary to construct the location program to any point that might be reached (say to record a difference). This can be done by adding onto the location program in Y15 (or Y16) the incremental location program from the initial location down to where Y11 (or Y12) currently is. Y19 is used for this and holds the incremental location program. Since the location program is only rarely desired, compared to all the movement back and forth over expressions, this location program is kept in reverse order so that it can be modified by push-down and pop-up operations. Thus, the Q2x routines put P8's and P9's into Y19 when going deeper and remove them when coming up. (Carrying out a simple example will make these considerations clear.) Routines exist which aid in the manipulation of Y19 (Q10, Q18).

The routine for comparing two corresponding cells belongs to the TE (F1 for K70, M23 for M19). This same routine is repeated for each pair of cells. It outputs the difference symbol applicable to the pair (into Y18). Since several differences may be applicable, this routine contains within it implicitly the order of importance of the differences. A flow diagram for F1, the comparison for symbolic logic (K70), is shown in Fig. 7.

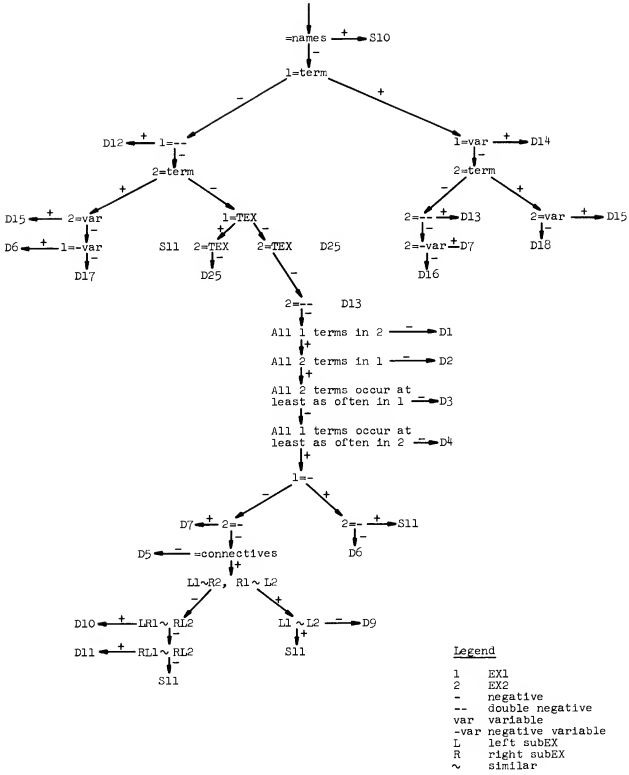


Fig. 7 F1: Compare EX1 and EX2.

R21 Match

The R20 match involves an implicit double iteration over the expressions. One iteration is employed by the scan represented by the Q2x's. But if the comparison routines are to see differences such as differences in terms (D1, D2, etc.), they must independently scan over the entire subexpression each time they are applied. The R21 match is an attempt to eliminate this feature by scanning over the expression just once, picking up information from each of the nodes, and then assembling the effective difference from the scraps of partial information. R21 still takes almost as much effort as the double iteration. However, it is useful in other ways; e.g., making it easy to see certain multiple and complex differences. Consequently it is the one normally used.

Figure 8 shows the flow diagram for R21, which is very similar to that for R20. The important differences are that when a difference is found (S12) a new signal is set (S16) before exiting. When this signal is seen by the main discrimination, data on the difference is recorded in list of difference expressions (in Y84). Q90 then resets the signal to S10 and the match proceeds (even though a difference has been found). The use of S16 allows termination of R21 when a difference is found, and the higher R-routines are to regain control. Subsequent reentry of R21 to continue the match is then possible. By the time the entire expressions have been scanned, all the differences that have been found are recorded on the list in Y84. Then, instead of quitting with S10 (as R20 does), R21 executes Q92, whose job is to analyze all the differences on the difference expression list and determine a single effective difference.

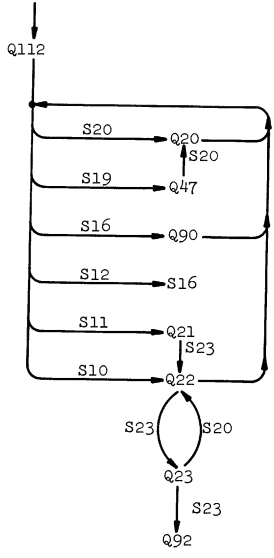


Fig. 8 R21: Match with Single Pass Getting List of Difference Expressions.

Combination of Differences: Q92

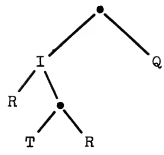
Q92 consists of a series of scans over the difference expression list, each time determining which pair of differences should be combined into a difference at a higher level. This is continued until only a single difference expression is left, which is then the effective difference for the match.

An important feature of the R21 match is that it scans the expressions only down to the point where a difference is found. At that point, the entire subexpressions are described on the difference expression. Besides the difference symbol (A89), the relative location program (the one determined by Y19) is created (A88) and lists are made of the terms in the subexpressions (A86 for #1, A87 for #2). These lists are marked to indicate the terms of each that are held in common (P44). This is done by putting S1 behind each term found on both lists. The term difference (if any) that exists at this location can be found from these lists (P45, yielding D30, D31, D32, D33, D34). Figure 9 shows two matched expressions and the resulting list of difference expressions.

In the main loop of Q92 the first part consists of determining two difference expressions to be combined. Notice in Fig. 9 that the difference expressions are on the list in order, so that adjacent difference expressions on the list designate adjacent differences in the tree. The basic act of combination is to take two differences and form them into a single difference at the lowest point in the tree that covers both of them (i.e., at their join, in lattice terms). Q92 is only prepared to combine pairs of differences, which implies that it must pick two differences whose join does not include any other differences. A way of doing this is to combine only adjacent differences whose levels constitute a

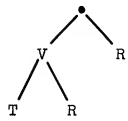
EX1

[RI(T.R)].Q



EX2

(TVR).R




```
Y84/  3725.                                  Cell holding list

3725/ 0                                      List of difference expressions
      9-1                                                (DE's)
      9-2

9-1/  9-10.                                  DE for lefthand side
9-10/ 0
      A89
      D32                                    D32 = delete from EX1
      A88
      P10                                    P10 = location program "left"
      A87
      9-11   9-11/0    9-12/0                Lists of terms
      A86          R          T
      9-12.        S1         S1
                   T          R
                   S1         S1.
                   R.

9-2/  9-20                                   DE for righthand side

9-20/ 0
      A89
      D34                                    D34 = disjoint terms
      A88
      P11                                    P11 = location program "right"
      A87
      9-21   9-21/0.   9/22/0                Lists of terms
      A86          Q.         R.
```

Fig. 9 Two Matched Expressions.

relative low. Thus Q92 moves along the list comparing successive triples; if it finds two that can be combined, it does so and starts over.

Combination involves creating a new difference expression whose location program (A88) is the join of the two components. The lists of terms for the join are determined afresh and compared against each other for term identity (P44).

The difference of the combination can be determined from some simple rules. If at least one of the component differences is not a term difference, then there is no interaction between the differences, and the difference of the join is simply the most important component difference, as determined by the difference ordering (e.g., by C10). If both component differences are term differences (D31 - D34) then interaction is possible if the difference in terms vanishes at the level of the combined difference (D30 from P45). For this means that the same set of terms is involved overall, but differences occur in subexpressions because of their arrangement. This implies either a position difference (D9) or a grouping difference (D10, D11), the exact inference depending on the type of differences of the subexpressions.

THE MATCH METHOD FOR TRANSFORM GOALS (K40)

A transform goal (K1) is defined as, "finding a way to transform object #1 (G1, G11) into object #2 (G1, G12)." The objects are given by both a TEX and a location program. Thus, G1 obtains TEX #1 and G11 locates the EX it is desired to transform. The transform goal has no specific output; its result is the sequence of operator applications that resulted in getting from #1 to #2. This is embedded in the goal tree. (Under some conditions, it would be appropriate to build a data structure of the operators that were used.)

The Method

The main method for attaining a transform goal is by the match method (K40). This method consists in matching #1 to #2 and, if a difference is found, setting up a subgoal of reducing this difference (K2). If no difference is found, then the two objects are already the same. If this reduce goal is successful, then a modification of #1 is produced and the subgoal is set up to transform the modified expression into #2. Thus, the method attempts to divide the total goal into two subgoals: one takes an initial step and the other attempts to go the rest of the way. The method consists of a list of segments, the separation between segments corresponding to major decisions to be made by the executive:

```
K40/ 0
     R30      Match and produce subgoal if difference exists
     Q28      Create and modify transform goal
     Q116 0   Set output
```

Match #1 to #2: Segment R30

The flow diagram for R30 is given in Fig. 10. It consists of a setup routine (Q25), which sets up the Y-cells from information on the goal (Y11-Y13-Y15-Y45, Y12-Y14-Y16-Y46, Y84-Y48, Y17); a major loop through a discrimination list; and a cleanup routine (Q24), which erases all the structures that have been created but not made into official structures and cleans out Y11, Y12, and Y19, which can have symbols stacked in them.

Immediate Operators

The initial signal set by Q25 is S19; this triggers the match (1Y91). The match can result in numerous signals. If S10 occurs, the two objects have been found to be identical, and it is only necessary to reset the signal

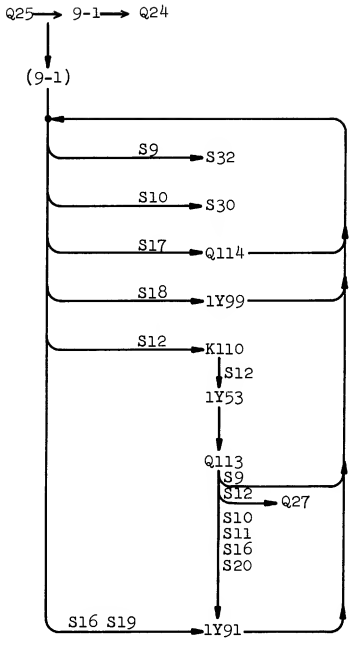


Fig. 10 R30: Match G1 to G2,
If Not Match Produce K3 (Reduce) Subgoal.

to S30, indication that the goal is achieved. If S12 occurs, a difference has been found. This is not necessarily the end of the line, since there exist immediate operators which might be applied to eliminate differences. An immediate operator is a routine that GPS can apply to take care of a difference. These may be part of the core (K110) or part of the TE (C3 or M3, in Y53). For example, C3 looks like:

```
C3/  I2
     D14
     Q52
     D15
     Q51
     D18
     Q53
     D21
     Q53.0
```

C3 is a branch on the difference (I2, which inspects Y18); if the difference found is D14, Q52 is executed, etc. Q51 and Q52 are substitution operators, corresponding to differences between a variable and an expression. Thus, if GPS sees a variable opposite an expression (D14, D15) it will immediately substitute for it. Q53 is a routine which resets the signal to indicate that matching is impossible (S13). For example, D18 is the difference of two terms in logic (e.g., P versus Q); when this occurs there is no way to transform P into Q, and so GPS should stop this attempt immediately rather than expend a large effort simply to conclude that one letter cannot be turned into another.

The immediate operators, if they occur, may change the signal (e.g., S12 to S10 for a successful substitution, or S12 to S13 for Q53); if they cannot correct the difference they will leave the signal S12. Q113, which follows 1Y101 in R30, is just a bookkeeping operation that records the final signal on the difference expression.

Create Subgoal

After the immediate operators have been tried, if a difference still exists (S12), then the reduce subgoal is created (Q27) and the segment is finished. Control returns to R11 which attempts the new subgoal, rejects it, etc. If the signal indicates that the difference is taken care of (S10, S11, S16, S20), then the match routine (1Y91) is re-entered and matching continues from where it left off. The match routine will continue in different ways depending on which signal occurs: S10 says the subEX's are identical, so go back up a level to continue matching; S11 says the subEX's are the same at this level, but the expressions need to be explored so go down a level to continue matching; S20 says there may be more comparison needed at this level; S16 says the provisional difference (in R21) still exists, so record it and continue matching.

Rematching

Two other signals are currently possible in the loop of R30. S17 indicates that something has happened, say because of the application of immediate operators, so that what was assumed no longer holds. The result is to start the match all over again (Q114 is another setup operation). The necessity of S17 arises because several differences may be discovered in a match; say, two occurrences of variables (two D15's) which require substitution, but happen to be the same variable. Action taken on them sequentially without exploring the consequences of intermediate actions causes trouble: the first substitution removes both variable occurrences and thus the second substitution cannot work.

Difference Selection

The other possible signal is S18, which indicates that a set of differences has been obtained, from which one must be selected (1Y99). This occurs in R21 after Q92 has finished. It is possible, when R30 is executed, that the two expressions have already been matched previously and as a consequence, the list of differences already exists on the goal (G53). In this case Q25 provides S18 and R30 immediately selects another difference, rather than going through the work of matching again.

Currently R30 can terminate with S30, S40, S42, or S32. This last implies failure in the attempt, and is used to summarize all the various ways the attempt could fail.

Create Modified Transform Goal: Segment Q28

If R30 supplies a reduce difference subgoal (K3) and R11 attempts it and succeeds, then R11 will execute the next segment of the method, Q28. The subgoal has obtained a result (G2, G13 on the subgoal) which can be used to build a transform goal to get from that result to the same final expression as the current goal (G2, G12 of the current goal). Q28 builds up this goal, tests to see if the newly constructed goal is identical to one already existing in the system (Q46 in Q17 in Q28), and turns the new subgoal over to the executive for action.

Final Segment: Q116

If R11 attempts the modified transform goal and succeeds, then it executes the final segment of the K40 method, Q116. This routine simply finds the correct signal (at A40 of the record of the most recent attempt) to indicate to the higher goal the final result. It is

necessary to obtain the signal from the goal, rather than having it available in a Y-cell, because it is unknown what might transpire between the attempt to obtain the modified transform goal and the use of this result by the supergoal.

THE TRY OPERATOR METHOD FOR APPLY GOALS (K41)

An apply goal (K2) is defined as "applying an operator (G15, G5) to an object (G1, G11)." The goal has a specific output (the first symbols in G3, G13), which is a new expression. (Recall that existing expressions cannot be modified.)

The Method

The method for applying an operator is again a list of several segments:

```
K41/ 0
     R31     Try operator; if fail produce difference subgoal
     Q38     Create modified applied goal
     R33.0   Produce output (Q29 or Q103)
```

The method has separate parts for each of the various types of operators. For the main type — the form operator — it matches the input (as the #1 expression) against the condition form of the operator (as the #2 expression). If this is successful, then the information so gained can be used to produce a new, modified expression from the produce form. If this is not successful, then a difference goal is set up. If this difference goal is attained, an apply subgoal is created using the modified expression provided by the reduce goal. In the case of operators with more than one input, this scheme requires an essential extension, which applies one of the component condition forms to the input expression and sets up a goal to find other suitable inputs from the

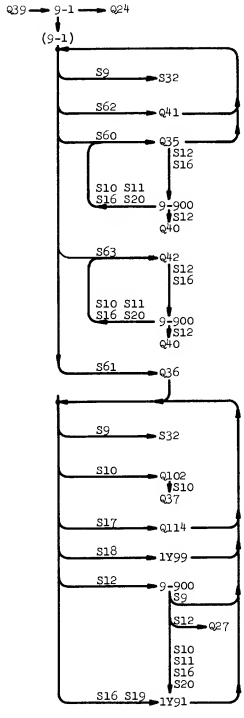
derivation list of the input expression for the additional components' condition forms.

Discriminate Type of Operator: Segment R31

The flow diagram for R31, the main segment, is given in Fig. 11. It divides roughly into two parts. In the upper part, there is a discrimination on the type of operator (A1) being applied. If the type is S60, there is a side condition to be applied (Q35, using a routine at A10). If this test fails, there may be a difference (S12), an attempt at immediate operators (9-900), and the creation of a reduce subgoal. If the side condition is satisfied, then the operator may still be of any other type and the discrimination is repeated. An operator may also be given directly by a routine (S63), in which case it is tried (Q42 from the routine at A11); again there is the possibility of a difference. An operator may be given by an expression (S62); for example, "reverse of operator X." In this case the actual operator is obtained (Q41) and then it is processed. Finally the operator may be given by a condition form and a product form (S61); this is the case GPS is set up to handle in detail and leads to the lower half of the R31 flow diagram.

Form Operators with One Input

This lower half is very similar to R30, consisting of a match, the use of immediate operators, the selection of differences, and so on. It differs from R30 in the action to be taken if a match is achieved (S10). R31 matches the expression (G1, G11) as #1 against the condition form of the operator (G5, G15 or P10 if G15 does not exist) as #2. The purpose is not only to see if the conditions are met, but to gather information



In all other cases method fails

Find operator given by expression

Test for operator applicability

Try immediate operators

If still difference, set up subgoal

Try direct operator

Try immediate operators

If still difference, set up subgoal

If a form operator, set up for match

In all other cases method fails

If match, prepare output if product undetermined

Produce product

If something has changed, rematch

If have DE's, select one

If difference, try immediate operators

If still difference, set up subgoal

Match

Try GPS immediate operators

If difference, try TE immediate Operators

Record result

Fig. 11 R31: Try Operator, If Not Work Produce K3 (Reduce) Subgoal.

in order to form the product; i.e., to identify the values of the variables. Hence Q37, which produces the output object, occurs after S10.

Form Operators with Two Inputs

GPS has to deal both with operators that have one input and with operators that have two inputs. In the former case, once the input is accepted (match in R31) the output can be produced (Q102 does nothing and S10 remains). However, if the operator has two inputs then even though the first input has been accepted, a second input is needed before the output can be produced. Two solutions to this problem are possible. First, the input to the operator is defined to be a single thing; i.e., a pair of objects. Thus, an attempt to apply a two-input operator to a single object reveals a "single vs. pair" difference, which can trigger a process for creating a pair. This solution was tried (Q55) and has been abandoned. The second solution is to permit the two-input operators to be applied to a single object, by deciding with which of the two input forms the object will be identified. The result, if successful, is a partially specified operator. This can be created as a new operator; it now only has a single input (the "other one") and it can be applied to various objects to see if a final result can be produced. In particular, it can be applied to all the objects on the derivation list. This attempt to apply an operator to a set will result in a "set vs. single" difference (D19), which will result in a selection of one of the objects on the derivation list.

The mechanics of this are somewhat involved. Two-input operators have a list (A17) which consists of pairs of symbols: location programs to their different input forms, followed by a cell for the name of the expression

which is accepted for this form (blank to begin with). As each of these input forms (there may be more than two) is used, its spot in the A17 list is filled and Q102 selects the next form to be filled. The first one is selected on the basis of trying the (two-input) operator against a single object, thus getting a "single vs. set" difference (D20), which results in the selection of one of the input forms as most similar to the object.

Although the total action depends on the other segments of the apply method, we give below a diagram of a typical application of a two line rule. (B25 is (AIB, BIC)Y(AIC); L stands for left subEX, R for right subEX.)

Goal 1: Apply B25 to PIQ

Goal 2: Reduce D20 between PIQ and input set

Select: LL B25

Goal 3: Apply LL B25 to PIQ

Produce operator: 1: (PIQ, QIC)Y(PIC)

Goal 4: Apply LR 1 to Deriv. list of PIQ = (SV(QIP), SVR, QIR)

Goal 5: Reduce D19 between set and LR 1

Select: QIR

Goal 6: Apply LR 1 to QIR

Produce object: 2: PIR

Although it appears that a good many steps are required to get through a single straightforward application of a two-input rule, it will be seen that the various selections, etc., are necessary.

Create Modified Apply Goal: Segment Q38

The second segment of the apply method (Q38) is analogous to Q28 for the transform-method. It sets up

the modified apply goal after the preceding reduce goal has provided a modified expression. The only difference is that Q38 must be prepared for the result to be either an object (K161) or an operator (K162). In the latter case, Q38 must set up a different modified apply goal in which the result becomes the new operator. (See example above.)

Final Segment: R33: Transferring Result (Q29) or Creating New Apply Goal (Q103)

The final segment (R33) either executes Q29, if the result is an object (S30, S36), or Q103, if the result is still not completely specified (S131, S132). Q29 transfers the results of this goal to be the result of the supergoal; e.g., the result of Goal 6, PIR, is also the result of Goal 1. Q103 creates another apply goal which applies one of the still undetermined input forms to the equivalence list of the original object; e.g., Goal 4 above. It then sets the signal for repeating the step (S46); this guarantees that Q103 will be executed enough times to get all of the input forms determined.

THE FIND RELEVANT OPERATOR METHOD FOR REDUCE GOALS (K42)

A reduce goal (K3) is defined as, "reduce the difference (G4) between an object (G1, G11) and a second object (G2, G12)." The goal has a specific result, a new object (the first symbols on G3, G13), which is a modification of object #1 (G1, G11). This object should not differ from the #2 object with respect to the specified difference (G4), although there is no guarantee of this. Likewise there is no guarantee that new differences have not been introduced.

The Method

The K42 method for reducing differences is a list of two segments:

```
K42/0
    R32     Find relevant operator and set up apply goal
    Q29.0   If subgoal successful, transfer result to
                                                this goal
```

Find Operator: Segment R32

The flow diagram for the main segment, R32, is given in Fig. 12. It consists of a setup (Q30), followed by the bulk of the program, followed by the standard clean-up routine, Q24. Q30 finds the list of relevant operators in the table of connections. There are two tables, one for GPS generally, which contains differences such as D19 and D20 (K59), and the other for the particular TE (in Y52). The tables of connections are description lists with differences as attributes and lists of relevant operators as values. Q30 will use the TE list if it exists (setting S69); if not, it will use the GPS list (setting S63); and if neither exists, it will set S2. Besides the table of connections, Q30 also sets up the #1 and #2 components, the various filters (see below), and the list of operators already tried (G30).

The body of R32 consists of one part for S63 (direct operators) and another for S69 (general operators). The distinction reflects the fact that GPS core has operators that are directly executed programs (1Y20). In the more general situation, it is necessary to go through the steps of setting up a subgoal and trying it through R31 (ultimately).

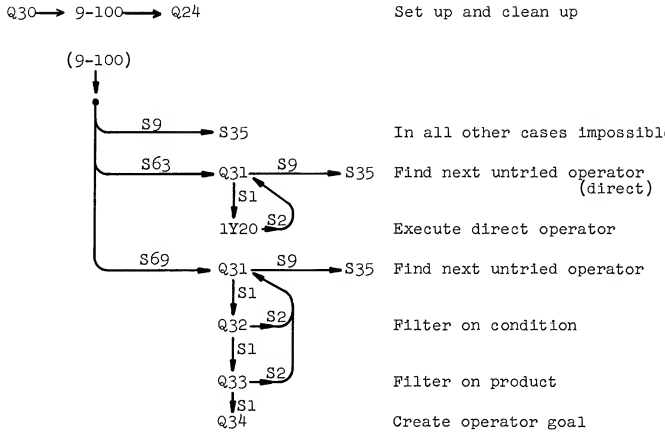


Fig. 12 R32: Find Next Untried Relevant Operator and Produce K2 Goal.

Find Next Untried Operator

Q31 is used to find the next untried operator by getting the next operator from the list of relevant operators and checking to see if it has been used before. There is no assumption that operators will be used in order; therefore, this check involves a J77 test on the list of used operators (in Y21 from G30).

Filters

There are two opportunities to test whether a proposed operator should be set up in a goal (usually called filters). The first involves testing for feasibility, the second for desirability (this latter has not yet been used). Examples of feasibility tests used in logic are tests for identical main connectives, or for size similarity. If the operator passes these preliminary tests, it is set up as an apply goal (Q34). The output is then S40, S42, S54 depending on Q46 in Q17 of Q34.

Transferring Result: Segment Q29

As usual, the executive takes the output of the segment, and if it indicates a subgoal, decides whether or not to attempt it. If it does and the result is favorable, then the next segment (Q29) simply makes the object produced by the subgoal (and available as the first symbols on G3, G13) the result of the reduce goal.

Repeatability of Method

If the apply subgoal fails (S32, etc.) then the executive decides whether to retry the method or to do something else. In the flow diagrams normally given for the methods, failure to produce a modified expression leads to a loop back to obtain another operator. Consistent with the general philosophy, this decision is up

to the executive (R10). Thus, methods are labeled (at A30) as either "not repeatable" (S80) or "repeatable" (S81).

THE TRANSFER EQUIVALENT RESULT METHOD FOR ALL GOALS (K43)

As described in the section on goal identity test, we distinguish duplicate goals (S54), which are of no use to GPS, and equivalent goals (S42), which are potentially valuable. In both cases the goal is identical (in the sense of defining attributes: G1, G11, G2, G12, G4, G5, G15, and G21) to some other goal already created. In the case of equivalence, the identical goals serve different purposes, and it may be profitable to share results obtained on one of the goals with the other (or others).

Single Segment: Q70

The mechanics of this are initiated in Q71, as already described; the borrowing is carried through by the K43 method, which consists of only a single segment, Q70. This method gets initiated because Q71 placed K43 on the method list of the goal with status S50. By the time Q70 is executed, Q71 has also already created a list of equivalent goals. This is available to each of the member goals (at G38). This common list contains on its description list, a list of all results generated to date by all the goals put together (A14). Each result consists of two parts (G3 and G13) and is packaged as a two-element list. On the description list of the equivalence list there is for each goal (as attribute) the name of the last result that was transferred to it from the common pool. Q70 gets one more result from this list and transfers it to the goal for which the K43 method is being executed. It then resets the marker so that this result will not be transferred again.

Blocking the Method

A problem in the use of this method is to avoid the continual checking for new results when none exist. Consequently, when a goal has received all the results available, Q70 changes the method status from S50 (or S51) to S54. This blocks the method from being used further. When a new result is added to the result list (A14) Q6, in recording this, goes to each of the member goals and changes the S54 back to S51.

V. THE EXPERIMENTER

INPUT CONVERSION AND SETUP

The experimenter executive is the first routine executed (currently E2). It first does a number of miscellaneous setups (E13); then converts and sets up the TE, which is given in Z90 (E23); then converts and sets up the top goal, which is given in Z91 (E22). At this point, it is ready to have GPS attempt the goal (1Y94). Following this a number of lists are printed and erased (L10 for goals, L11 for TEX's, L12 for goal equivalence lists).

Set Up Trivia: E13

E13, which does all the miscellaneous initialization, first sets up the signals and attributes. These items each have a common form: a signal Sx is of form 10Sx.I8; an attribute Ax is of form 10Ax.J10; and an attribute Gx is of form 10Gx.L, where L links to a routine (see E15) that will bring the goal back in from auxiliary storage if needed. Lists of the symbols to be made into signals and attributes (L5 for signals, L6 for attributes, L15 for goal attributes) are fed to E10 along with their forms (E13, E17, E15). Initially this was done to avoid writing each routine separately; it has since proved of advantage in changing the action taken by attributes. (It also permits new signals and goal attributes to be defined by GPS, but this feature has not been exploited.)

E13 next takes an input list of identifications to be made (L1). This list consists of pairs, say X Y, which may be read "make X identical to Y." This is accomplished by a full word store in which X receives the same PQ SYMB LINK as Y.

Next, E13 sets up a number of things for output (see also Output). The routines named on input list L3 are marked with Q=3 for tracing; the symbols on input list L4 are marked with Q=4 for propagating trace; the symbols on input list L7 are fixed to "trace" by putting their names in the signal line (E19); and the symbols on list L18 are given the output names associated with them on L18 (by E16, at A19).

Finally E13 modifies the TE (in Z90) by putting on it the pairs given on list L17. This is either an addition or a replacement depending on whether the value is new or already exists on the TE.

TE Conversion

E23, which converts and sets up the TE, uses Q79 to put the TE symbols into the Y-cells. It then constructs a composite list of variables, adding the list from GPS (in K56) to the list from the TE (in K82). Finally it takes the TE operators (on list in Y51) and the TE objects (on list in Y54) and converts them to internal form (E21).

Goal Conversion

E22, which converts an externally given goal, first checks to see if the goal is in internal form (since we wish to allow a complete goal list structure to be put in from outside). The indicator is the existence of the goal type at attribute G21; in the external form this is given in the first list cell. If the goal needs conversion, a form is obtained at A20 of the goal type, copied, and established as the basic description list of the goal. Then the various components of the goal are converted. This requires a division of the routine according to goal type, since the format of information on

the external goal list depends on goal type. Again, conversion of objects is done with E21, so that the objects for goals need not have been previously converted.

TEX Conversion: E21

The most complex initial conversion is from the external form of a TEX, which is a linear list, such as (A V B) = (B V A), to the internal form, which is a tree structure. This is handled by E21 in two parts. Each TE has its own external format, and hence the conversion of TEX itself is done by a TE routine (in Z80). Beyond this, however, there are several things to be done in common for all TE objects. If a location program is given externally, then this should be recorded at A9, as well as being put at Y81. The TE must be recorded at A12. If the object is an expression for an operator, the operator must be produced. If the object is an operator with a set of inputs, it requires a list to keep track of what objects are assigned to which component input forms. This is obtained by copying a form (K97) and attaching it at A70. Finally, if a TEX is a set of objects, then its subobjects should be set up as TEX's, and not just as EX's (P43).

Conversion of Parenthetical Expressions

The conversion of parenthetical expressions, although specific to the TE and accomplished by a TE routine (F10 for logic), is of common enough occurrence that central processes are available out of which specific conversion routines can be built. A set of cells (Z40 to Z47) is assigned purely to conversion processes and a set of routines (E30 to E41) provides component routines. A basic assumption is that the input list (to be converted) will be a linear list, consisting

of a finite known alphabet of significant characters (i.e., those that signal some action in the conversion process), plus additional characters which are simply to be transferred. Thus the organization of the conversion routine is in the form of description lists, with characters as attributes and conversion action programs as values. Any symbol which is not on this list is taken over into the converted expression without change. These lists are loaded into Z30 by the conversion routine and interpreted there by E31. This permits the interpretation to change as a function of the conversion process (e.g., F10 uses one list for converting the description list, another for converting the logic expression). Another assumption is that the converted expression will have the same name as the original; hence E30 removes the head from the input list and establishes it as the first cell of the converted expression (Z41, Z42, Z43). The initial list, now called the working list, is saved in Z45 for later erasing (E39) and put into Z44, which acts as a running pointer to it (being advanced by E32).

Throughout the conversion process it is necessary to keep a pushdown list of P8's and P9's (in Z46) out of which a location program can be fashioned if a character is encountered that requires it (E36, which puts it in Z47). This list, as well as other stacks that might be built up during the conversion process, are all cleaned up by E39.

Several composite routines are available which accomplish large portions of a conversion. E37 takes the next step in a conversion where parentheses have their usual meaning, either transferring a symbol or, if the next symbol is "(", creating the sublist to the matching ")" and transferring its name as the next symbol. E38 makes a sublist out of the remaining symbols on the

input list. E40 simply follows the basic cycle of interpret and advance. E41 is a conversion of a parenthetical expression into a list structure, leaving all other symbols unchanged.

OUTPUT AND DEBUGGING

The output of all runs is a trace of the behavior of the program. A run is shown as Appendix A. This is a clean run without any tracing for debugging purposes; if there had been some it would have been intermixed at the point of its occurrence in the run.

The first page is the spec sheet, which will be discussed in the next section on setting up a run. The second page is the problem-solving attempt proper. This is followed by the Post-Mortem, which is not shown.

Behavior Trace

Each goal is printed out in full the first time it is attempted (E24 in Q81). The level in the goal tree is first given followed by the name (at A2, which is the order of generation number taken from Y34); then the defining phrase; then the supergoal; and finally the internal name of the list structure (for debugging purposes). The integer at the far right gives the cycle count, H3. The occurrence of this expression indicates that GPS is now attempting this goal.

The course of the program's behavior can be followed by the "signal trace"; i.e., the lines of signals and other symbols occurring throughout the run. Each time a signal is discriminated (by I1, I2, I3, or I4), it is put into the print line (by E70 in Z92). If the discrimination is made by I11, I12, etc., then the signal is not recorded in the print line. In addition, whenever certain routines occur (those recorded on L7),

they record their own name in the print line along with parentheses; this makes it easy to group the signals in terms of the subroutines in which they occur. Also, whenever GPS passes into the context of a goal (other than a new one), it records the goal name in the print line (E25). With these items and the flow diagrams of the R-routines it is possible to trace what GPS did through a run.*

Every new object that is created is also printed out (E26 or E68 in P50). This includes its name (at A2, which is the order of generation from Y36), the object according to its format as given by the TE routine, and the internal symbol for the list structure (for debugging purposes). In addition, various other major decisions of the program rate special messages: goal rejected, operator rejected (E27), goal selected (E69), object too complex, and so on.

Printing Formats

The printing of all the expressions and statements is handled in a uniform way. There is a print format consisting of a list of information to be printed across the page. This format is interpreted by E50, which reads each symbol of the format and loads the print line. The rules E50 follows are: if the symbol in the format list is an alphabetic data term, it is entered as the next chunk of information to be printed; if the symbol is not an alphabetic data term, it is assumed to be a "format routine" and it is executed. The format routines take their inputs in HO. The main one, E57, is used to record the external name of (O) in the print line: if (O)

*See Sec. II, "A Tour Through a Simple Problem."

is a data term, the data term is printed; if (0) has a value at A19 this is taken as the name; if it has a value (integer) at A2 then this is used for the name; finally, if none of these hold, the symbol itself is taken as the name. Besides E57 there are several others: E54 and E55 for advancing the column number; E56 if (0) is a format; E61 if (0) is a list of names, E63 if (0) is a location program (which might involve substituting a special expression); and E64 if (0) is a difference (which might involve substituting a special expression). E50 uses several subroutines (E51, E52, E53) and some standard cells (Z50, Z51) to perform its task.

Besides E50 there are a few additional print routines: E58 to print a simple list, and E59 to print a "linear" list. The latter is a special form that comes from the execution of E60 on a list structure. E60 creates a linear parenthesized form of a structure, consisting of nothing but alphabetic data terms.

Debugging Facilities

Besides the ability to trace any routine selectively, as provided for in the IPL manual, a few additional debugging facilities are provided. The most useful is a collection of "monitor points" that are built into various routines where experience has shown it is desirable to be able to execute an arbitrary monitoring routine. These are the Z9x cells, each of which gives in its title the routine in which it is executed (e.g., 1Z95 executed in E25). The IPL Post-Mortem on the 7090 executes the routine in W14 after it has executed the rest of the Post-Mortem (a GPS patch). E12 is the standard routine used to give the contents of various cells plus the prints of a few lists. The final debugging aid consists of two routines which will trace a routine if

and only if it occurs in certain goal contexts (i.e., the number at A2 of 1Y2). E11 specifies the goal; E18 specifies an interval of goals.

SET UP FOR RUNNING

Assemblies and Modifications

In general, runs are made from an assembled version on tape. A typical run consists of reading the total system in from the tape; loading some additional permanent routines and data, either new ones or modifications of old ones (J165); saving the updated system on tape (J166); loading some additional routines and data that are unique to this run and temporary; and then kicking off with E2.

Spec Sheet

There is always at least a page of assembled information unique to the run to specify the various parameters and lists. This is called the spec sheet and is shown as the first page of the run in Appendix A. Most of the individual cells are self-explanatory. Z90 holds the TE; Z91 holds the task; i.e., the top goal. The rest of the Z9x cells hold the monitor routines. This is followed by some Y-cells (Y90 - Y101) which hold the names of various important routines in GPS-Core. All the various lists for tracing and modifying (L1, L3, etc.) also occur here. Finally there are a few data terms, such as Z7, the available space limit for reading goals to auxiliary storage; K32, the time limit in cycles (H3); and K34, the limit on goal depth.

AUXILIARY STORAGE

There is automatic storage of goals onto auxiliary storage when space becomes scarce. Every time a new goal is created (Q16) routine E7 (file goals if available space is less than Z7) is executed. If it is necessary to get more space, E7 starts at the top goal (1Y111) and attempts to file each goal. Certain exceptions are made: if the goal is closely related to the current context (in Y2, Y7, Y9, Y10, Y87, Y88, Z28); if the goal is already on auxiliary (A8); or if the goal has been marked to stay in core (A7). A goal to be filed is split into two parts: all those attributes named on list L29 are kept in core with the goal; all the rest are moved to a separate structure, which is then filed (J107). The head cell of this structure (which is now the auxiliary control word) is kept on the goal at A8. After E7 has filed all the goals it can, available space is rechecked; if space is still shy, the signal S139 is recorded and the run is terminated.

Whenever it is necessary to work with a goal (Q80), E8 is executed to bring the goal back into core if it was on auxiliary. This action simply undoes what E7 did: the structure at A8 is moved in (J105) and it is merged with the goal structure that was left in main storage. Besides the transitions from one goal context to another, there are also occasions to examine a few features of a great number of goals, such as in the process of selecting which subgoal to try next (Q109). Thus, every time a goal attribute (Gx) is executed and its value not found, it is necessary to determine if that goal is on auxiliary and, if so, to bring it in. This is automatically handled by the goal attribute routines (from form E15).

VI. TASK ENVIRONMENTS

SYMBOLIC LOGIC TE (K70)

Types of Information

This TE gives the necessary information to do the kind of problems used by O. K. Moore.[15] It consists of a set of operators (B1 to B25), given as forms (e.g., AVB = BVA); a set of objects (B50 to B99), which are logic expressions (e.g., (-P.Q)V(P.-P)); various lists of objects and constants (C1 to C9); the table of connections (C2); a set of differences (D1 to D39); the ordering of the differences by difficulty or importance (C10, C11); and a collection of routines for various functions (F1 through F32). All of these items of information are obtained either directly or indirectly through K70.

Differences and Associated Structures

The differences have no information associated with them directly. They function purely as selective intermediates: they are produced by the comparison routines during the match (F1, F24, F26), and are used to select lists of operators on the table of connections (C2). The relations between them are given by the ordering of differences (C10, C11). These latter consist of a list whose items are either difference symbols or lists of difference symbols. There is a routine (P48) to test if a difference is on the ordering (some, such as D19 and D20, are not). There is also a routine (P7) to compare two differences. This outputs a signal: S4 if (0) occurs before (1) in the list; S5 if (0) and (1) are the same difference or occur in the same sublist of differences; and S6 if (0) occurs after (1) in the list.

Several TE routines implement the comparison of two EX's in Y11 and Y12. Recall that the match is divided into two parts: the putting of two structures into correspondence, done by GPS-core; and the comparison of two such pieces, done by the TE routine. The latter is the one that detects and assigns the differences. These comparisons are discussed in detail in the section on matching and will not be repeated here. There are several comparison routines (F1 for R20; F20, F24, F26 for R21) reflecting attempts to fit GPS to different protocols. Several of the tests for specific differences have been centralized into routines (F30, F31, F32), just to make modifications easier.

Multiple Negation Signs

Several of the TE routines (F4, F5, F6, F7) deal with the manipulation of multiple negative signs. According to the rules of logic used in these problems, a positive sign may be freely replaced with a double negation and vice versa. This is actually expressed in the rules by talking of "sign changes," but is realized in GPS-2-2 by doing sign manipulation by means of immediate operators (see C3). The problem of signs is also reflected in handling substitutions. Thus, in the system of logic used here if -A is opposite PVP (where A is a variable), then it is possible to substitute A = -(PVP) to produce identity. This requires an adjustment in the substitution routines (F25 and F27, which use some elementary operations; F28 and F29).

Double negations occur mostly through the act of substitution. The format of expressions makes it difficult to become aware of a double negation when it is formed. Hence the philosophy has been to carry them along until they are spotted as a difference (D6 or D7)

during some later match. At this time, an immediate operator (F6 or F7) would get rid of them. It has also proved convenient, mostly for output purposes, to go over the TEX and remove all multiple negation signs at once (F9).

Filters and Similarity Tests

The final part of the routines provides the various operator conditions, filters, and similarity tests which cannot be expressed by forms (F2, F3, F8, F21, F22, F23).

MISSIONARIES AND CANNIBALS TE (M19)

This TE gives the necessary information to enable GPS to work on the Missionaries and Cannibals puzzle. There are three missionaries and three cannibals on one side of a river, with a boat that holds two people. All six can row the boat. The problem is to get all six people to the other side of the river without ever letting more cannibals than missionaries exist on either side of the river, in which case the missionaries would be eaten. The cannibals are sufficiently reliable, however, to be trusted to row the boat by themselves or stay on one side of the river by themselves.

Types of Information

The TE consists of a set of operators (M30 to M39); a set of differences (M40 to M59); a set of objects (M70 and M71); a single problem (M80); a set of lists of various items (M1 to M9); a table of connections (M2); an ordering of differences (M10); a set of routines (M20 to M28); and a set of symbols for handling the side condition about more missionaries than cannibals (M11, M60 to M63, M90 to M93).

Most of these entities are strictly analogous to those in the symbolic logic TE (K70): the table of connections, the difference ordering, the identity test, the filters, etc. The differences reside in the way of handling the operators (M22), the admissibility test (M27, etc.), and a special executive used for some runs (R2 involving M28). The basic format for objects has already been discussed in the section on TEX's. Likewise, M22 and the format for operators was illustrated in detail in the section on Operators. Neither of these will be discussed further here.

Admissibility Test

The admissibility test (M27) is built to take as input a symbol (M90 - M93) which designates which side is to be checked. It checks the indicated side by putting the missionaries and cannibals in one-to-one correspondence and emits either "satisfied" (S1) or an "unsatisfied" (S2), along with an indication of the failure (M60 to M63 in Y18). There are four ways of specifying the side to be checked: the left side, which initially holds all the men (M90); the right side, which must hold all the men at the end (M91); the side from which the men are moving on this boatload (M92); and the side to which the boatload is moving (M93). The reason for the different ways of designating sides comes from attempts to simulate human subjects, who have a tendency to check, say, left side and "to" side. Since these sometimes designate only a single side between them, they can lead to failure to observe the admissibility constraint. There is a list (M11) which gives the set of admissibility tests to be applied.

External Task Space: Top Executive R1

In running GPS on the M&C problem and comparing it with human performance, it was observed that the humans often did not remember any of the intervening positions. They knew the initial position, they knew the position they were at, and they knew the position they had just come from. This contrasted with the situation in logic where intermediate positions were known and often used. Part of this difference rests in the different external arrangements of the tasks. In logic (K70) the "official" results of applying operators were kept on the blackboard in plain view. (There could, of course, be other results which the subject though of but never made explicit.)

In M&C the subjects worked with a graphical representation of the river, using physical objects for missionaries and cannibals and physically moving them from one side to the other. As a consequence, they had no external memory of any position but the current one. Top executive R1 is an attempt to simulate this latter situation. A list of external TEX's (L13) is kept, which simulates the external graphical device. For recording purposes this contains all the TEX's ever obtained in order, but for GPS the only TEX that counts is the last one on the list, which represents the current situation. GPS must command the experimenter to apply an operator to this current position (done by executing M28 rather than by a communication). M28 applies the operator to be the last TEX on L13, if it is feasible, and adds a new TEX to the end of L13. It then applies a complete set of admissibility tests and if the move is not admissible another move is made that undoes the move (thus leaving a graphical record of the failure to satisfy the rules). Two new signals are used in the problem-solving

executive: S44 (final problem solved) which occurs if the final state is ever obtained by the TEX on L13, and S45 (external progress made) which occurs whenever a new TEX is generated on L13 (excluding the returns to previous TEX). When S44 occurs R10 is able to quit. Whenever S45 occurs, R1 sets up a new transform goal to get from the new TEX to the final result. R1 is written in a reasonably general fashion; however, it has never been tried with any task except M&C.

ADDING NEW TASK ENVIRONMENTS

The addition of a new TE still requires an intimate knowledge of the way GPS works and of its internal conventions. To install a new TE, the best course is to take one of the existing TE's and ask what the relevance of each part is to the new task. To illustrate, suppose we wanted GPS to work on trigonometric identities--a task that has been hand simulated in detail,[5] but not run yet on GPS. We consider K70, the TE list for symbolic logic, as providing the most appropriate check list.

```
K70/0
    K81         Difference ordering
    C10
    K82         List of variables
    C6
    K84         Difference print list
    C19
    Z80         Convert TEX
    F10
    Z81         Print TEX
    E26
    Y51         List of operators
    C1
    Y52         Table of connectives
    C2
    Y53         List of immediate operators
    C3
    Y54         List of objects
    C4
    Y60         Identity comparison
```

```
P20
Y62        Similarity test for objects  sets
F21
Y63        Compare objects
F1
Y64        Compare operators
F1
Y65        Search filter on operator conditions
F22
Y69        Standardization
F9
Y70        Similarity test for operator sets
F23
Y72        Adjustment for EX1 (Q51)
F25
Y73        Adjustment for EX2 (Q52)
F27.0
```

Many of the considerations below are obvious and parallel to information already given. Nevertheless, it is useful to have it all in one place, oriented toward introducing a new TE. To be simple-minded, we will take up the items as they occur on K70.

Difference Ordering: K81

The set of difference symbols (Dx) are not yet selected, but will be later. A look at C10 shows that we can assign differences to indifference subclasses if we wish. In any event we will undoubtedly need a special list for Trig (call it T10). The function of the ordering is to permit the evaluation of goals.

List of Variables: K82

For simple problems in trigonometry, we will only need variables in the algebraic operators (like A^2-B^2 = (A+B).(A-B)). We might as well use A, B, C, D, which are used in Logic; hence C6 can be used directly.

Difference Print List: K84

This structure is used to make the output pretty. When a difference symbol is going to be printed in a format, this list is consulted and if format information exists for the difference symbol, it is used. Thus, "REDUCE D9 ON L1 TO L0" becomes, "CHANGE POSITION ON L1 TO L0." There is no need to develop such a list for a new TE at the outset, so we will just leave it out.

Convert TEX: Z80

For most new TE's the input format is quite idiosyncratic to the task area and an entirely new routine has to be thought through. For Trig, however, we can work quite close to Logic, since they both use parenthetical notations. An examination of F10 shows that there is an interpretation of the various symbols in the input line, each calling forth its own conversion subroutine. Thus V, ., I are treated as binary operators. Trig demands a symbol for equality, addition, subtraction, multiplication, division, and exponentiation. Suppose we use =, +, -, *, /, and ** for these. We incur a few additional problems in the minus sign, which must be admitted as a unary operator when it occurs initially; and in the exponentiation sign, which has a double symbol, so that the decision on whether * or ** has occurred requires some memory or a forward and backward look. Since equality is part of the object expression connectives, we need another symbol, say E, for the operator connective. In any event, a routine (U10) can be written using F10 as a model that will convert an expression in standard external notation into the accepted internal structure.

Multiple Operands

A much more difficult problem will be encountered if it is desired to use addition and multiplication as operations with an indefinite number of operands; e.g., $\sin x + \cos x + \sin^2 x + 1$. It is easy enough to code the conversion routine to give a list with + as the head and all the operands in the list cells. The problems arise in getting operators to work on expressions with indefinite operands. The kind of form GPS knows about--e.g., A*(B+C) = A*B+A*C--assumes fixed structure. To get GPS to work with A*(B+...+D) = A*B+...+A*D requires some additional ingenuity. Even to apply a binary operator anywhere within a set of operands of indefinite length requires ingenuity. Since the purpose here is to illustrate, rather than solve new problems, let us agree to stick with operations with fixed operands. (This is what was done in the hand simulation.)

Print TEX: Z81

A look at E26, the Logic print, shows it depends on E60, which produces a linear string of characters from a tree structure. Examination of E60 shows that it, like F10, examines each of the important symbols in the input structure to select a specific subroutine to build the linear list. A routine is needed for Trig that can easily be modeled on E60 (say U60). However, E60 is not directly named in the TE list. Either a variant of E26 must be written using U60, or perhaps a new TE cell can be created (say Y79) such that E26 uses the routine in Y79 and the TE list specifies what routine it should be. (The TE cell for printing cannot be eliminated, since not all TE's can use E26.)

List of Operators: Y51

Most of the operators for Trig can be written down right away. The algebraic ones are clear; so are the trigonometric identities--e.g., $\sin^2 x + \cos^2 x = 1$. This may be expressed as two rules, one running in each direction; GPS does not yet recognize rules as two sided. (However, the second rule can be expressed simply as the reverse of the first.)

However, one operator--the combine operation--cannot be expressed as a form. It simplifies expressions by recursively applying to them a whole series of rules: 0 + 0 = 0, 1*n = n, $x^1 = x$, $x^0 = 1$, x/x = 1, x + x = 2*x, etc. This operator should be coded as a direct operator (A1 = S63)--that is, as an IPL program. This will be a rather extensive piece of code; it was coded once for GPS-1 in IPL-IV and ran about 200 instructions. In coding this operator, it can be assumed that the operand is in Y11-Y13-Y15. Alternatively, of course, the component laws could be expressed as separate forms, letting the simplification emerge from the general attempts at a solution. Some sort of general trend toward simplicity, as expressed by additional differences, might be required.

Numerical Calculation

A subsidiary problem, but one that is quite important for mathematical manipulation in general, is how to represent numbers and get numerical calculations carried out. Some form of rational arithmetic is required, such that 2/2 = 1 and 4/2 = 2, but 3/2 = 3/2. In Trig this only shows up in the combine operator; still it requires an agreement on representation, on when operations will be performed, etc.

Table of Connections: Y52

The form of the table of connections can be seen from C2. For each of the differences there must be a list of what operators are considered relevant and in what order. Clearly a new structure T2 is needed for Trig.

List of Immediate Operators: Y53

Again the form of the list can be seen from C3. For Logic it consists of ways of handling the double negation, and a response to two different constant terms (e.g., P versus Q) that the difference is impossible to reduce (set S13). The immediate operators for D14 and D15 that accomplish substitution are associated with GPS-core, since they are part of the general apparatus to apply form operators. For Trig, according to the way we were proceeding above, the sign is to be a binary operation (with some unresolved problems about the initial sign). This means we do not want the D6, D7, D12, D13 immediate operators, and their function will be taken over by the combine operator. On the other hand, it may be easier to still think of '-' as an unary operator. This would change the conversion routines so that x-y becomes x + (-y); then these immediate operators of Logic would perhaps be appropriate. One can see from this that basic decisions about representation can affect everything else.

In a similar vein, if we handle the trigonometric functions as constants (since we are ignoring their arguments completely for simple problems), then there exist ways of transforming one term into another via trigonometric identities. Hence we do not want the D18 immediate operator. On the other hand, if we continue to represent "sinx" as an expression with sin as the

operation and x as the operand, then the D18 immediate operator is still reasonable.

There is probably little point in trying to think of additional immediate operators until some runs show where something is needed. If we take the right options above, we don't need any immediate operators at all, and can just leave Y53 off the list.

List of Objects: Y54

For Logic, C4 is an empty list, since the only objects used are those defined in the top goal and objects derived from them. Y54 could just as well have been left out. The same is true of Trig. However, any objects put on the object list will be converted.

Identity Comparison: Y60

This test is used when a new object has been created to find if an identical TEX already exists. P20 is a general comparison of two list structures, ignoring the description lists. It should be perfectly suitable for Trig.

Similarity Test for Object Sets: Y62

This routine is directed toward selecting out the one member of a set of objects that is most similar to an external object. It occurs in Q54, the direct operator that is evoked by D19. For example, if the initial problem is given as getting from a set of objects to a specified one, then matching produces D19, which ends up by selecting the most similar one of the set as the starting point. The similarity test used for logic, F21, demands that the main connective (read operation, for Trig) is the same and that the two objects have at least one term in common. If used in Trig, it would tend to

classify sums with sums, products with products, etc., and would call expressions dissimilar if they didn't both contain the sin, or the cos, etc. This probably isn't exactly the right shape for Trig, but it is a good start.

Compare Objects: Y63

This is the part of the match routine that is task independent. The two expressions are put into correspondence on their structure and the routine in Y63 is executed at each pair of subexpressions. The output of the comparison routine is one of the signals: S10 (the entire subexpressions headed at this point are identical); S11 (no differences at this point, but the subparts of the subexpressions need investigation); S12 (a difference exists at this point); or S13 (it is impossible to make these two expressions the same). In the latter two cases the difference symbol (Dx) is placed in Y18.

This IPL routine provides half the operational definition of the differences, the rest being provided by the table of connections. The routine for Logic, F1, is diagrammed in Fig. 7. Many of the same differences will be appropriate; in fact, perhaps F1 could be used to get started for Trig. New differences can be introduced by expanding the compare routine to output a new symbol when it detects some new feature, and associating some operators with the symbol on the table of connections.

Compare Operators: Y64

Conceivably a difference comparison should be used when trying to satisfy the condition form of an operator rather than comparing two objects. This has not proved to be the case for Logic, and initially there is no reason to suppose it true for Trig.

Search Filter on Operator Conditions: Y65

In selecting an operator to reduce a difference, a preliminary selection is made on the feasibility of the operator. GPS will run perfectly well without any filter; and one can be added later. Again however, the ones for Logic (F22 or F8) are good candidates for useful ones for Trig.

Standardization: Y69

This routine was introduced into Logic to remove all the double negation signs prior to printing the object. There is no reason to consider such a routine for Trig until the need becomes manifest.

Similarity Test for Operator Sets: Y70

This routine is analogous to the similarity test for objects, Y62. It occurs in connection with Q56 and D20, which is related to the two-line rules. Since there are no two-line rules in the operator set for Trig, there is no need for this.

Adjustment for EX1 (Q51): Y72

This routine permits GPS to see a negative variable as a variable--that is, if -A is opposite PVQ, then -(PVQ) is substituted for A. The immediate operators take care of whatever double negations occur. Whether something like this is needed for Trig depends intimately on the issues mentioned earlier about how to handle signs.

Adjustment for EX2 (Q52): Y73

This routine is analogous to the one for Y72, but concerns variables in EX2 rather than variables in EX1.

Summary

We have now covered the range of items in the Logic TE list. We have raised some representational issues plus the question of how to get arithmetic done. These may require a good bit of thought before they can be satisfactorily settled. There are several substantial routines to code: the conversion, the print, and (at some time) the compare. However, the Logic routines provide good models. There may be entirely new TE dependent elements, but none of these are apparent yet. Hence, if we assume the analysis above, we can build a new list for Trig, say K71:

```
K71/0
    K81
    T10       New difference ordering
    K82
    C6
    Z80
    U10       New conversion
    Z81
    U26       New print, like E26, but using U60 in
    Y51                                  place of E60
    T1        New list of operators
    Y52
    T2        New table of connections, but with Logic
    Y60                                      differences
    P20
    Y62
    F21
    Y63
    F1        Use Logic compare to get started
    Y64
    F1
    Y65
    F22.0     Use Logic filter to get started
```

Besides K71 and the structures that are named on it, we must write down all the operators and code up the combine operator. Also, we must write down a few objects and put their names on goals. Finally, we are ready to assemble all the new routines, put K71 in Z90 and the goal name in Z91, and GPS will attempt a problem in the

Trig task environment. In fact, over and above the bugs in the new programs and structures, there are sure to be a few conceptual errors that will require modification of the TE, including perhaps the addition of new routines.

The chosen example, trigonometry, was almost guaranteed to be easy, since it is so similar to Logic. If we had picked a quite different task, say chess or various puzzles, we would have been faced with a much more intricate problem of how to represent the essentials of the task so that they fit GPS's way of doing thing.

Appendix A

OPS RUN ON "R.(-PIQ) INTO (QVP).R"

```
2704    0 0 1024   0          Z90          K70   0          TASK ENVIRONMENT
2705    0 0 520    0          Z91          C36   0          TASK
2706    0 0 734    12594      Z92          E70               SIGNAL MONITOR
12594   0 0 735    0                       E71   0
2710    0 0 26258  0          Z96          J0    0
2714    0 0 26258  0          Z100         J0    0          MONITOR INPUT PRINT FORMAT
2494    0 0 2014   0          Y90          R10   0          PROBLEM SOLVING EXEC
2495    0 0 2024   0          Y91          R20   0          MATCH
2496    0 0 1868   0          Y92          Q74   0          EVALUATE NEW SUBGOAL
2497    0 0 1598   0          Y93          P14   0          MAKE VARS DISJOINT
2498    0 0 2006   0          Y94          R2    0          TOP EXEC
2499    0 0 1870   0          Y95          Q76   0          CONSTRUCT GOAL VALUE
2500    0 0 1903   0          Y96          Q109  0          SELECTION OF NEW GOAL IN R10
2501    0 0 1899   0          Y97          Q105  0          SELECTION OF SUBGOAL FROM LIST
2502    0 0 1674   0          Y98          P90   0          DESCRIPTION OF MARKED LIST
2503    0 0 1904   0          Y99          Q110  0          SELECTION OF NEXT DIFF
2504    0 0 1898   0          Y100         Q104  0          EVALUATE EQUIVALENT SUBGOAL
1135    0 4 0      0          L1     +     0                LIST OF IDENTIFICATIONS
1137    0 4 0      0          L3     +     0                Q=3 TRACE LIST
1141    0 4 0      12593      L7     +     0                LIST OF ROUTINES FOR SIGNAL ( )
12593   0 0 2014   12595            +     R10
12595   0 0 2015   12596            +     R11
12596   0 0 2024   12597            +     R20
12597   0 0 2025   12598            +     R21
12598   0 0 2034   12599            +     R30
12599   0 0 2035   12600            +     R31
12600   0 0 2036   12601            +     R32
12601   0 0 2037   0                +     R33   0
1151    0 4 0      12602      L17   +     0                LIST FOR TE MODIFICATION
12602   0 0 2467   12603            +     Y65
12603   0 0 775    12604            +     F1
12604   0 0 2468   12605            +     Y64
12605   0 0 775    0                +     F1    0
986     0 1        1000000    K32   + 0 1   100   0000     ABSOLUTE EFFORT LIMIT
988     0 1              20   K34   + 0 1         20       ABSOLUTE DEPTH
1037    0 0 1055   0          K83   +     K101  0          GOAL VALUE TYPE
691     0 0 26258  0          E27          J0    0          TEMP SUPPRESS PRINT
```

L0 QVP .R (883) 35654

L1 R. -PIQ (882) 36269

1 GOAL 1 TRANSFORM L1 INTO L0 (SUBGOAL OF NONE)(C36) 36833

28 L1 DERIVATION LIST 37816

29 L0 DERIVATION LIST 38690

30 R1 DERIVATION LIST 39730
(R10. S50 S50 (R11. S50 (R30. S19 (R20. S19 S20 S12) S12 D9 S12 D9 S12) S40 S8

2 GOAL 2 REDUCE D9 ON L1 TO L0 (SUBGOAL OF 1)(13633) 44750
(R10. S50 S50 (R11. S50 (R32. S69 S1 S2 S1 S1 S11 S40 S8

3 GOAL 3 APPLY R1 TO L1 (SUBGOAL OF 2)(13711) 47080
(R10. S50 S50 (R11. S50 (R31. S61 S19 (R20. S19 S20 S11 S20 S12) S12 D15 S10 S10 (R20. S10 S20 S12) S12 D15 S10 S10 (
R20. S10 S23 S20 S23 S23) S10 S10

L2 -PIQ .R (13796) 52738
) S30) S301 GOAL 2 S30 S48 S41 S30) S30) GOAL 1 S30 S48 S41 S40 S8

2 GOAL 4 TRANSFORM L2 INTO L0 (SUBGOAL OF 1)(137901 54953
(R10. S50 S50 (R11. S50 (R30. S19 (R20. S19 S20 S11 S20 S12) S12 D5 S12 D5 S121 S40 S7

3 GOAL 5 REDUCE D5 ON P9 L2 TO P9 L0 (SUBGOAL OF 4)(13929) 60301
(R10. S50 S50 (R11. S50 (R32. S69 S1 S2 S1 S2 S1 S1 S11 S40 S8

4 GOAL 6 APPLY R6 TO P9 L2 (SUBGOAL OF 5)(140191 63020
(R10. S50 S50 (R11. S50 (R31. S61 S19 (R20. S19 S20 S11 S20 S12) S12 D15 S10 S10 (R20. S10 S20 S12) S12 D15 S10 S10 (
R20. S10 S23 S20 S23 S23) S10 S10

L3 PVQ .R (14109) 68608
1 S30) S30) GOAL 5 S30 S48 S41 S30) S30) GOAL 4 S30 S48 S41 S40 S8

3 GOAL 7 TRANSFORM L3 INTO L0 (SUBGOAL OF 4)(14139) 70826
(R10. S50 S50 (R11. S50 (R30. S19 (R20. S19 S20 S11 S20 S12) S12 D9 S12 D9 S121 S40 S6

4 GOAL 8 REDUCE D9 ON P9 L3 TO P9 L0 (SUBGOAL OF 7)(14214) 76721
(R10. S50 S50 (R11. S50 (R32. S69 S1 S1 S11 S40 S8

5 GOAL 9 APPLY R1 TO P9 L3 (SUBGOAL OF 8)(143051 78584
(R10. S50 S50 (R11. S50 (R31. S61 S19 (R20. S19 S20 S11 S20 S12) S12 D15 S10 S10 (R20. S10 S20 S12) S12 D15 S10 S10 (
R20. S10 S23 S20 S23 S231 S10 S10

L4 QVP .R (14404) 84111
) S30) S30) GOAL 8 S30 S48 S41 S30) S301 GOAL 7 S30 S48 S41 S40 S8

4 GOAL 10 TRANSFORM L4 INTO L0 (SUBGOAL OF 7)(143801 86329
(R10. S50 S50 (R11. S50 (R30. S19 (R20. S19 S20 S11 S20 S11 S20 S10 S20 S10 S23 S20 S20 S10 S23 S20 S23 S23) S10)

S30) S301 GOAL 7 S30 S48 S41 S301 S30) GOAL 4 S30 S48 S41 S30) S30) GOAL 1 S30 S48 S41 S301 S301

Appendix B

GPS-2-2 VOCABULARY (ROUTINES)

```
GENERAL ATTRIBUTES
  COMPONENT ATTRIBUTES
    A1     TYPE OF COMPONENT
    A2     ORDER OF GENERATION (NAME)
    A3     GOAL THAT PRODUCED COMPONENT
    A4     LIST OF GOALS USING COMPONENTS
    A5     LIST OF EQUIVALENT COMPONENTS
    A7     MARK TO KEEP IN CORE
    A8     CONTROL WORD FOR AUX STORAGE
    A9     LOCATION PROGRAM, IF EXTERNAL
    A10    TEST FOR OPERATOR CONDITIONS
    A11    DIRECT PROGRAM FOR OPERATOR
    A12    TE OF OBJECT
    A13    LIST OF VARIABLES FOR TEX
    A14    RESULT LIST (EQUIVALENT GOALS LIST)
    A15    COMPLEXITY OF TEX (0) (INTEGER)
    A16    MAX DEPTH OF TEX (0) (INTEGER)
    A17    DIRECT PROGRAM FOR COMPARISON
    A19    CHARACTER TO BE PRINTED


  GOAL TYPE ATTRIBUTES
    A20    GOAL FORM


  METHOD ATTRIBUTES
    A30    METHOD TYPE


  RESULT ATTRIBUTES
    A40    ATTEMPT STATUS
    A41    LOCATION OF METHOD USED
    A42    STATUS OF METHOD
    A43    START OF SUBGOAL ATTEMPT
    A44    FINAL SUBGOAL


  GENERAL ATTRIBUTES
    A51    CONTENT TYPE


  LOCATION PROGRAM TREES
    A60    LOCATION PROGRAM
    A61    LEVEL
```

MISCELLANEOUS ATTRIBUTES
A70 LIST OF COMPONENTS USED (MULT. OPR)

DIFFERENCE EXPRESSION ATTRIBUTES
A80 STATUS
A81 NBR 1 SUB DE
A82 NBR 2 SUB DE
A83 EVALUATION TYPE- K10X
A84 LEVEL (ABSOLUTE)
A85 MAX - LEVEL (ABSOLUTE)
A87 NBR 2 LIST OF OCCURRENCES
A88 RELATIVE LOC PROGRAM
A89 DIFFERENCE TYPE
A90 LIST OF DIFFERENCE TYPES
A109 HIGHEST DEFINED REGIONAL

EXPERIMENTER ROUTINES
EXECUTIVES
E2 TOP EXECUTIVE FOR SINGLE TASK
E7 FILE GOALS IF SPACE LESS THAN Z7
E8 MOVE GOAL (0) IN FROM AUX

MISCELLANEOUS
E10 DEFINE ONE WORD ROUTINES
E12 POST MORTEM PRINT
E13 INITIAL SET UP OF TRIVIA
E14 FORM FOR SIGNAL
E15 FORM FOR ATTRIBUTE
E16 SET PRINT NAMES (A19) A/C LIST (0)
E19 SET SUBR (0) FOR SIGNAL-LINE TRACE

INPUT - OUTPUT ROUTINES
E21 CONVERT TEX (1), CONTENT TYPE (0)
E22 CONVERT GOAL (0) TO INTERNAL FORM
E23 CONVERT TE (0) TO INTERNAL FORM
E24 PRINT GOAL EXPRESSION
E25 PRINT GOAL NAME AND DEPTH
E26 PRINT TEX (0)
E27 PRINT OPERATOR REJECTED
E28 PRINT GOAL NAME
E29 PRINT GOAL STRUCTURE

```
BASIC CONVERSION ROUTINES
  E30   SET UP FOR CONVERSION
  E31   INTERPRET CURRENT SYMBOL
  E32   LOCATE NEXT INPUT SYMBOL
  E33   CREATE SUBLIST (NOT CONNECTED)
  E34   CREATE NEXT CELL
  E35   RETURN TO LOC IN PRIOR SUBLIST
  E36   FIND LOC PROG, PUT IN Z47 (SAFE)
  E37   CREATE NEXT UNIT, SYMBOL OR ()
  E38   CREATE NEXT EXTENDED UNIT
  E39   CLEAN UP CONVERSION (H5 SAFE)
  E40   FOLLOW INTERPRETATION LIST
  E41   CONVERT () TO LIST STRUCTURE
  E49   ASSIGN A PRINT NAME TO (0)


OUTPUT ROUTINES
  E50   PRINT STANDARD FORMAT (0)
  E51   SET UP FOR PRINTING
  E52   CLEAN UP PRINTING
  E53   LOCATE NEXT CELL IN FORMAT
  E54   ADVANCE COL NUMBER 1 SPACE
  E55   ADVANCE COL A/C DEPTH (0)
  E56   PRINT SUBFORMAT (0)
  E57   ENTER NAME (0)
  E58   ENTER SIMPLE LIST OF SYMBOLS (0)
  E59   ENTER LINEAR LIST (0)
  E60   CREATE LINEAR LIST FOR EX(0)
  E61   ENTER LIST OF NAMES (0)
  E62   ENTER DIFFERENCE EXPRESSION
  E63   ENTER LOCATION PROGRAM
  E64   ENTER DIFFERENCE (0)
  E68   PRINT EQUIVALENCE LIST (0)
  E69   PRINT GOAL SELECTED
  E70   ENTER SIGNAL (0) IN PRINT LINE
  E71   TEST IF XO IN HO
  E109  HIGHEST DEFINED REGIONAL


LOGIC TE
  PROCESSES
    F1    DIFFERENCE NET
    F2    OPR APPL TEST, EX 1 MAIN, POSITIVE
    F3    OPR APPL TEST, EX 1 MAIN
    F4    ADD A DOUBLE NOT TO EX 1 (KNOWN +)
    F5    ADD A DOUBLE NOT TO EX 2 (KNOWN +)
    F6    REDUCE MULTIPLE NOTS ON EX 1
    F7    REDUCE MULTIPLE NOTS ON EX 2
    F8    OPERATOR FILTER
    F9    STANARDIZE EX(0) (DOUBLE NOTS)
    F10   CONVERT TEX (0) TO INTERNAL
    F20   DIFFERENCE NET, LOCAL
    F21   SIMILARITY TEST ON TEXS (0), (1)
    F22   OPR FILTER, NBR 2
    F23   SIMILARITY TEST FOR OPR SETS
    F24   DIFF NET, TEST IF EX2 - TERM (D5)
```

F25 ADJUST EX 1, VAR 2 FOR SUBS
F26 DIFF NET, TEST IF EX2 - TERM (D9)
F27 ADJUST EX 2, VAR 1 FOR SUBS
F28 ADD A DOUBLE NOT
F29 ADD A DOUBLE NOT
F30 TEST IF CONNECTIVE DIFF (D5)
F31 TEST IF POSITION DIFF (D9)
F32 TEST IF LOWER SIGN DIFFS EXIST (D8)
F39 HIGHEST DEFINED REGIONAL

GOAL ATTRIBUTES
COMPONENTS
G1 TEX 1
G2 TEX 2
G3 LIST OF RESULTS
G4 DIFFERENCE
G5 OPERATOR

ADDITIONAL COMPONENT SPECS
G11 LOC PROGRAM FOR EX 1
G12 LOC PROGRAM FOR EX 2
G13 LIST OF LOC PROGRAMS FOR RESULTS
G15 LOC PROGRAM FOR OPERATOR

MISCELLANEOUS
G20 GOAL STATUS
G21 GOAL TYPE
G22 CURRENT VALUE
G23 SUPERGOAL
G24 LIST OF SUBGOALS
G25 LIST OF SUBGOAL TRIES
G26 HISTORY OF ATTEMPTS
G27 METHOD LIST
G28 METHOD USED TO GET THIS GOAL
G29 LOC OF SEG USED TO GET THIS GOAL
G30 LIST OF OPERATORS TRIED
G31 TE FOR THIS GOAL
G32 ACTION FOR SEGMENT
G33 GOAL NET (TOP GOAL ONLY)
G34 LIST OF VARIABLES USED
G35 ANTECEDENT GOAL
G36 RESULT STATUS
G37 LIST OF ATTEMPTS THE GOAL PART OF
G38 LIST OF EQUIVALENT GOALS
G39 LIST OF UNTRIED LOWER GOALS

```
  MEASURES
    G40   ABSOLUTE DEPTH


  MISCELLANEOUS
    G50   G1-EXPANDED GOAL
    G51   PERMANENT LIST, S50-S51 SUBGOALS
    G52   GOAL NBR AT LAST TRY (FOR G50 GOAL)
    G53   COMPOUND DIFFERENCE EXPRESSION
    G54   MOST RECENT ATTEMPT STATUS
    G109  HIGHEST DEFINED REGIONAL


ADDITIONAL SYSTEM ROUTINES
  INTERPRETATION ROUTINES
    I1    INTERPRET SIGNAL IN Y1 (S9 IF NONE)
    I2    INTERPRET DIFFERENCE IN Y18
    I3    INTERPRET GOAL TYPE IN Y3
    I4    INTERPRET CONTENT TYPE IN Y85
    I8    LOAD SIGNAL FROM H0
    I9    FIND SIGNAL LIST, PLUG H1.
    I11   I1 WITHOUT MONITOR
    I12   I2 WITHOUT MONITOR
    I13   I3 WITHOUT MONITOR
    I14   I4 WITHOUT MONITOR
    I18   1Z92 I18
    I19   FIND SIGNAL INTERPRETATION,


  MISCELLANEOUS
    I20   SAVE FOR RESTART
    I21   SAVE ON SYSBR1
    I29   HIGHEST DEFINED REGIONAL


MISSIONARIES AND CANNIBALS TE
  PROGRAM
    M20   CONVERT TEX (0) TO INTERNAL
    M21   PRINT TEX (0)
    M22   GENERAL OPERATOR ROUTINE
    M23   DIFFERENCE NET
    M24   IDENTITY TEST (0), (1) (LOC MAIN)
    M25   FILTER OPR ON CONDITION
    M26   FILTER OPR ON CONDITION
    M27   TEST ADMISSIBILITY OF EX(0) (LOC)
    M28   COMMAND TEX 1Y80
    M109  HIGHEST DEFINED REGIONAL
```

```
P - ROUTINES
  MISCELLANEOUS
    P1     TALLY 1 AND RESTORE
    P2     SET (0) TO 0 AND RESTORE
    P3     SET EFFORT BASE INTO (0)
    P5     SUBTRACT 1 FROM (0), RESTORE
    P6     COMPARE (0) AND (1) (NUMBERS)
    P7     COMPARE (0) AND (1) ON ORDERING (2)
    P8     LOCATE MAIN EX, LOCATE NEXT EX
    P9     LOCATE FIRST SUBEXPRESSION


  EXPRESSIONS (INPUTS ARE LOCS OF EX)
    P10    LOC PROG FOR FIRST SUBEX (IN K98)
    P11    LOC PROG FOR SECOND SUBEX (IN K98)
    P12    EXECUTE EX(0). OUTPUT IS INPUT (0)
    P13    COPY TEX (0)
    P14    MAKE VARS OF (0) AND (1) DISJOINT
    P15    GENERATE LOC OF TERMS OF EX(1)
    P16    GENERATE LOC OF VARIABLES OF EX(1)
    P17    GENERATE LOC OF CONSTANTS OF EX(1)
    P18    GENERATE LOC OF SUBEX OF EX(1)
    P19    CREATE A NEW VARIABLE
    P20    COMPARE LIST STRUCTURES LOC (0),(1)
    P21    TEST IF (0) IS A VARIABLE
    P22    TEST IF (0) IS A SET
    P23    TEST IF (0) IS TEX
    P24    TEST IF TERMS OF EX(1) ARE ON EX(0)
    P25    TEST IF ALL CONSTANT TERMS OF EX(1)
    P26    TEST IF EX(0), (1) HAVE SAME TERMS
    P27    TEST IF (0) FIRST LEVEL EX OR TERM
    P28    TEST EX(0) HAS MORE TERMS THAN EX 1
    P29    TEST IF EX(0), (1) HAVE COMMON TERM


  OPERATOR ROUTINES
    P30    PRODUCE REVERSED OPR


  MISCELLANEOUS
    P40    FIND LEVEL OF ABS LOC PROG (0)
    P41    LOC 1ST ITEM LIST(2) THAT PASS COMP
    P42    DELETE ITEMS LIST(2) THAT FAIL COMP
    P43    IF OPR(0)=, MAKE OPERANDS INTO TEXS
    P44    MARK COMMON TERMS OF LISTS (0),(1)
    P45    DESCRIBE MARKED LISTS IN Y18
    P46    COMPARE LEVEL OF LOC PROGS (0), (1)
    P47    FIND LOCPRG OF JOIN LOCPRGS (0),(1)
    P48    TEST IF (0) ON ORDERING (1)
    P49    COMPARE VALUES (0), (1) (K101)
    P50    ASSIGN NAME TO TEX (0)
    P51    MAKE VARS OF (0) AND (1) DISJOINT
    P52    CREATE VAR LIST FOR TEX (0)   (A13)
    P53    COMPUTE COMPLEXITY OF TEX(0)   (A15)
    P54    COMPUTE MAX DEPTH OF TEX (0)   (A16)
    P55    COMPUTE MAX DEPTH OF EX(0)
```

```
    P56   GENERALIZED COMPARE (0), (1)
    P57   COMPARE VALUES (0), (1) (K102)
    P58   FIND(CREATE) EQUIV LIST FOR TEX (0)
    P59   COMPARE VALUES (0), (1) (K103)


  BASIC HOUSEKEEPING
    P60   COPY AND MAKE LOCAL
    P61   DELETE ALL SYMBOLS
    P62   IF LOCAL COPY LOCALLY, IF NOT NO-OP
    P63   GENERATE LOCATION OF LIST (0)
    P64   GEN LOCATIONS OF LIST (1), (2)
    P66   MOVE ATR-VALUE (0) FROM (1) TO (2)
    P68   POP HO TWICE
    P69   JO/0


  LOCATION PROGRAMS (IN K97, K98)
    P70   LOC 2ND SUBEX 1 LEVEL DOWN (K97)
    P71   ABS LOC PROG. LEFT OF LEFT (K98)
    P72   ABS LOC PROG. RIGHT OF LEFT (K98)
    P73   RELATIVE NEXT
    P74   RELATIVE FIRST SUBEX, 1 LEVEL DOWN


  EXPRESSIONS
    P80   LOC FIRST NON-UNARY EX IN EX(1)
    P81   FIND MAIN EX OF (0)
    P82   LOC MAIN GIVEN (0) = LOC OF TEX/EX


  MISCELLANEOUS
    P90   DESCRIBE MARKED LISTS, NBR 2
    P91   FIND LEVEL OF REL LOC PROG (0)
    P92   LIST HIGHEST VALUED DE-S IN DE(0)
    P93   TEMP FOR K70 ON S4 C32
    P109  HIGHEST DEFINED REGIONAL


Q - ROUTINES
  EXECUTIVE SEGMENTS
    Q1    WHAT LIMIT IS EXCEEDED
    Q2    LOCATE NEXT UNTRIED METHOD
    Q3    LOC NEXT SEGMENT AND EXECUTE
    Q4    EVALUATE GOAL
    Q5    TEST IF AT TOP LEVEL
    Q6    RECORD ATTEMPT (LEAVE SIGNAL)
    Q7    REPEAT CURRENT METHOD IF REPEATABLE
    Q8    SELECT BEST UNTRIED LOWER GOAL
    Q9    RETRY ANTECDENT GOAL
```

MISCELLANEOUS

Q10	FIND NEW LOC PROG=(0) + Y19 RECORD
Q11	COPY Y11-Y13-Y15-Y45 IF NEEDED
Q12	COPY Y12-Y14-Y16-Y46 IF NEEDED
Q13	ASSIGN (1) TO BE ATR (0) OF GOAL
Q14	ASSIGN (1) TO BE ATR (0) OF GOAL
Q15	ADD (1) TO ATR LIST (0) OF GOAL
Q16	COMMON FRONT PART OF GOAL CREATION
Q17	COMMON END PART OF GOAL CREATION
Q18	FIND REL LOC PROG FOR Y19 LIST
Q19	ASSIGN LIST ATRS (0) FROM GOAL (1)

METHOD SEGMENTS AND SUBSEGMENTS

Q20	COMPARE NBR 1 AND NBR 2 A/C 1Y17
Q21	FIND FIRST PAIR, 1 LEVEL DOWN
Q22	FIND NEXT PAIR, THIS LEVEL
Q23	RETURN ONE LEVEL UP
Q24	CLEAN UP Y11-Y20
Q25	SET UP MATCH FOR G1 TO G2
Q26	FIND IMMEDIATE TE OPR
Q27	CREATE DIFFERENCE GOAL
Q28	CREATE MODIFIED TRANSFORM GOAL
Q29	USE RESULT OF SBGL AS RESLT OF GOAL
Q30	SET UP TO FIND RELEVANT OPERATOR
Q31	FIND NEXT UNTRIED OPERATOR
Q32	FILTER OPERATOR ON CONDITIONS
Q33	FILTER OPERATOR ON PRODUCT
Q34	CREATE APPLY OPERATOR GOAL
Q35	TEST OPR COND FOR APPLICABILITY
Q36	SET UP OPR FOR MATCH (Q39 DONE)
Q37	CREATE PRODUCT FROM FORM
Q38	CREATE MODIFIED APPLY GOAL
Q39	FIND OPERATOR TYPE AND SET UP
Q40	CREATE DIFF GOAL FOR DIRECT TEST
Q41	FIND OPR GIVEN BY EXP, SET UP
Q42	TRY DIRECT OPERATOR
Q43	TEST IF NEW TEX ON EQUIV LIST
Q44	SET VALUES ON Y3X LIMITS
Q45	CREATE EQUIVALENCE LISTS
Q46	TEST IF GOAL ALREADY EXISTS ON 1Y25
Q47	FIND GPS DIFFERENCES
Q48	RECORDS FOR NEW TEX
Q49	RECORDS FOR SELECTED TEX

```
IMMEDIATE OPERATORS
  Q50   COMMUTE SET EX 1
  Q51   SUBS EX 1 FOR EX 2 (VAR) IN TEX 2
  Q52   SUBS EX 2 FOR EX 1 (VAR) IN TEX 1
  Q53   IMPOSSIBLE IF NOT PROVISIONAL
  Q54   SELECT FROM Y11 SET
  Q56   SELECT FROM Y12 SET


METHOD SEGMENTS AND SUBSEGMENTS
  Q70   TRANSFER A RESULT FROM EQUIV GOALS
  Q71   TEST IF GOALS EQUIV, AND SET UP
  Q72   CREATE ATTEMPT RECORD
  Q73   SELECT BEST SUBGOAL ON LIST (0)
  Q74   EVALUATE GOAL (NBR 2)
  Q75   OBJECT VARIATION METHOD FOR K1
  Q76   CREATE GOAL VALUE (K101)
  Q77   CREATE GOAL VALUE (K102)
  Q78   CREATE GOAL VALUE (K103)
  Q79   SET NEW TE (0)


GOAL SETTING ROUTINES
  Q80   GOAL SET ROUTINE, (0) = GOAL
  Q81   SET UP NEW SUBGOAL (IN Y87)
  Q82   SET UP SUPER-GOAL FOR RETURN
  Q83   SET UP GOAL FOR RETRY
  Q84   SET UP PRIOR GOAL FOR RETURN
  Q85   SET UP SUPERGOAL OF 1Y88 FOR RETRY
  Q86   SET UP SUBGOAL FOR RETRY
  Q87   SET UP SUPER-GOAL FOR RETRY
  Q89   RESET MTH-SEG CONTEXT FROM GOAL (0)


MATCH ROUTINES
  Q90   CREATE DIFF-EXP AND PUT ON 1Y84
  Q92   COMBINE LIST Y84 OF DES (NBR 2)


METHOD SEGMENTS AND SUBSEGMENTS
  Q100  RE-EXECUTE CURRENT SEGMENT
  Q101  SET METHOD ACTION SIGNAL
  Q102  PREPARE OUTPUT IF NOT DETERMINED
  Q103  CREATE NEXT OPR GOAL FROM SET
  Q104  GOAL REJECTED
  Q105  SELECT BEST GOAL FROM LIST (0)
  Q106  EVALUATE GOAL (NBR 3)
  Q107  SET COMPLEXITY LIMITS
  Q108  TRY G1-EXPANDED GOAL
  Q109  SELECT BEST S50, S51 LOWER GOAL
  Q110  SELECT NEXT DIFF FROM 1Y84
  Q111  SET Y1X CONTEXT FOR DE (0)
  Q112  TEST IF MATCH ALREADY DONE
  Q113  RECORD STATUS OF DE
  Q114  ERASE DE AND SETUP FOR REMATCH
  Q115  RECORD METHOD ATTEMPT
```

```
    Q116  SET OUTPUT FOR K40 METHOD
    Q118  ADD P8 TO Y19
    Q119  ADD P9 TO Y19
    Q209  HIGHEST DEFINED REGIONAL


R - ROUTINES (NO PREFIXES, NO YS)
  EXECUTIVES
    R1    KEEP WORKING FROM EXTERNAL
    R2    TOP EXECUTIVE FOR NEW PROBLEM


  PROBLEM EXECUTIVES
    R10   PROBLEM SOLVING EXECUTIVE
    R11   EXECUTE METHOD UNTIL FAIL


  MATCHES
    R20   MATCH DEPTH FIRST
    R21   MATCH, SINGLE PASS (Q90-Q92)


  METHOD SEGMENTS
    R30   MATCH G1 TO G2, CREATE SUBGOAL
    R31   TRY OPERATOR, CREATE SUBGOAL
    R32   FIND NEXT UNTRIED RELEVANT OPR
    R33   SECOND STEP IN K41 METHOD
    R109  HIGHEST DEFINED REGIONAL


SIGNALS (SX = 10SX/I8)
  GENERAL
    S1    YES, +, POSITIVE, OK, FIND, ACCEPT
    S2    NO, -, NEGATIVE, NOT FIND, REJECT
    S3    MUCH WORSE
    S4    SOME WORSE, LESS
    S5    THE SAME
    S6    SOME BETTER, GREATER
    S7    MUCH BETTER
    S8    UNDEFINED
    S9    NO INTERPRETATION


  DIFFERENCE RESULTS
    S10   IDENTICAL
    S11   NO DIFF SO FAR, MAY BE DEEPER
    S12   DIFFERENCE FOUND
    S13   HOPELESS, TOO DIFFERENT
    S14   OPERATORS DIFFER
    S15   OPERANDS DIFFER
    S16   PROVISIONAL DIFFERENCE
    S17   SOMETHING HAS CHANGED
    S18   DIFFERENCE EXPRESSION EXISTS
    S19   START OF DIFFERENCE
```

```
CORRESPONDENCE RESULTS
  S20   BOTH 1Y11 AND 1Y12 FOUND
  S21   1Y11 FOUND, 1Y12 NOT, DEEPER
  S22   1Y11 NOT, 1Y12 FOUND, DEEPER
  S23   NEITHER 1Y11 NOR 1Y12 FOUND, DEEPER
  S24   1Y11 FOUND, 1Y12 NOT, MORE OPERANDS
  S25   1Y11 NOT, 1Y12 FOUND, MORE OPERANDS
  S26   1Y11, 1Y12 HAVE SAME NBR OPERANDS


ATTEMPT STATUS
  S30   SUCCESS, ONE RESULT
  S31   SUCCESS, SEVERAL RESULTS
  S32   TRIED UNSUCCESSFULLY
  S33   UNTRIED
  S34   INCOMPLETE
  S35   IMPOSSIBLE (METHOD EXHAUSTED)
  S36   IDENTICAL RESULT
  S37   BORROWED, ONE RESULT
  S38   BORROWED, SEVERAL RESULTS
  S39   BORROWED, UNSUCCESSFUL


SEGMENT RESULTS
  S40   SUCCEED, NEW SUBGOAL GENERATED
  S41   SUCCEED, MORE SEGMENTS
  S42   SUBGOAL GENERATED (=ONE EXISTING)
  S43   SUBGOAL REJECTED
  S44   FINAL PROBLEM SOLVED
  S45   EXTERNAL PROGRESS MADE
  S46   SUCCEED, REPEAT SEGMENT
  S47   SUBGOAL FAILS
  S48   SUBGOAL SUCCEEDS


METHOD STATUS AND GOAL STATUS
  S50   UNTRIED
  S51   NOT THROUGH
  S52   THROUGH
  S53   BLOCKED
  S54   DUPLICATION


COMPONENT TYPE
  S60   OPERATOR WITH INITIAL CONDITIONS
  S61   FORM OPERATOR
  S62   EXPRESSION FOR OPERATOR
  S63   DIRECT ACTION OPERATOR (KNOWS Y'S)
  S69   GENERAL OPERATOR
```

```
LIMITS
  S70   ABSOLUTE NUMBER OF GOALS
  S71   RELATIVE NUMBER OF GOALS
  S72   ABSOLUTE EFFORT
  S73   RELATIVE EFFORT
  S74   ABSOLUTE DEPTH
  S75   RELATIVE DEPTH
  S76   ABSOLUTE NUMBER OF OBJECTS
  S77   RELATIVE NUMBER OF OBJECTS
  S78   ABSOLUTE NUMBER OF METHOD TRIES
  S79   RELATIVE NUMBER OF METHOD TRIES


METHOD AND GOAL TYPE PROPERTIES
  S80   NOT REPEATABLE
  S81   REPEATABLE


GENERAL
  S90   NONE
  S91   1
  S92   2
  S93   3
  S94   4
  S95   SOME
  S96   TERM
  S97   UNARY
  S98   FIRST LEVEL
  S99   COMPLEX
  S100  PERMANENT
  S101  TEMPORARY


ATTEMPT STATUS
  S130  SUCCESS, REJECT FOR COMPLEXITY
  S131  SUCCESS, STILL INDETERMINENT
  S132  SUCCESS, S131, IDENTICAL
  S139  OUT OF SPACE
  S209  HIGHEST DEFINED REGIONAL
```

Appendix C

GPS-2-2 VOCABULARY (DATA)

```
LOGIC TE OPERATORS AND OBJECTS
  OPERATORS (OKMOORE), RX ARE HIS NAMES
    B1     R1 AVB YIELDS BVA
    B2     R1 A.B YIELDS B.A
    B3     R2 AIB YIELDS -B1-A
    B4     R3 AVA = A
    B5     R3 REVERSE B4, A = AVA
    B6     R3 A.A = A
    B7     R3 REVERSE B6, A = A.A
    B8     R4 AV(BVC) = (AVB)VC
    B9     R4 REVERSE B8, (AVB)VC = AV(BVC)
    B10    R4 A.(B.C) = (A.B).C
    B11    R4 REVERSE B10, (A.B).C = A.(B.C)
    B12    R5 AVB = -(-A.-B)
    B13    R5 REVERSE B12, -(-A.-B) = AVB
    B14    R6 AIB = -AVB
    B15    R6 REVERSE B14, -AVB = AIB
    B16    R7 A.(BVC) = (A.B)V(A.C)
    B17    R7 REVERSE B16, (A.B)V(A.C)=A.(BVC)
    B18    R7 AV(B.C) = (AVB).(AVC)
    B19    R7 REVERSE B18, (AVB).(AVC)=AV(B.C)
    B20    R8 A.B YIELDS A  MAIN, POSITIVE
    B21    R8 A.B YIELDS B  MAIN, POSITIVE
    B22    R9 A YIELDS AVB, MAIN
    B23    R10 A,B YIELDS A.B
    B24    R11 AIB,A YIELDS B
    B25    R12 AIB, BIC YIELDS AIC


  OBJECTS
    B50    (-P.Q)V(P.-P)                          A1
    B51    Q                                      A1
    B52    P.(Q.R),-(RIT)I-(P.Q)                  A2
    B53    T.T                                    A2
    B54    PV(QVR),-(QVR).S2-P                    A3
    B55    -QI-S                                  A3
    B56    PVQ,-RI-Q,S,RI-S                       A4
    B57    PVT                                    A4
    B58    (P.-P).(RIT)                           B1
    B59    QVS                                    B1
    B60    (PVQ)I-(-RVP),-(-(S.Q)VR)              B2
    B61    -Q                                     B2
    B62    -PIQ,-RIQ,-PV-R                        B3
    B63    QVS                                    B3
    B64    (PVP)I-Q,QVR,RIS,P                     B4
    B65    (S.R)VT                                B4
    B66    (PVQ).(QIR)                            C1
    B67    PV(Q.R)                                C1
```

```
    B68   (P.Q)V(P.T),TI(P.R)                    C2
    B69   QVR                                    C2
    B70   -S,RVS,(PIQ)I-R                        C3
    B71   -Q                                     C3
    B72   PIQ,-RI(P.Q),QI-T,(P.Q)I-P             C4
    B73   -PV(-T.R)                              C4
    B74   (RI-T).(-RIQ)                          D1
    B75   -(-Q.P)                                D1
    B76   (PVQ)IR,RIS                            D2
    B77   -PVS                                   D2
    B78   (PIQ)I-R,RVS,-S                        D3
    B79   -Q                                     D3
    B80   (PVQ)I(RVS),P,-TI-(QVR),-T             D4
    B81   S                                      D4
    B82   R.(-PIQ)                         ALPHA 1
    B83   (QVP).R                           ALPHA 1
    B84   -S.(SV-Q.                         ALPHA 2
    B85   -Q                                ALPHA 2
    B86   (P.Q)V(P.-P)                      ALPHA 3
    B87   PIQ                               ALPHA 3
    B90   P,Q
    B91   P,PIQ
    B92   PIQ,QIR
    B93   AV(BVP) = (AVB)VP                 (COPY B8)
    B94   -(PVP)IR,-R
    B95   P
    B96   -PVQ
    B97   PIQ
    B98   Q
    B99   P.Q
    B109  HIGHEST DEFINED REGIONAL


LOGIC TASK ENVIRONMENT
  CONSTANTS, LISTS, PROBLEMS
    C1    LIST OF OPERATORS
    C2    TABLE OF CONNECTIONS
    C3    LIST OF IMMEDIATE OPRS FOR DIFFS
    C4    LIST OF OBJECTS
    C5    LIST OF EQUIVALENCE LIST OF OBJECTS
    C6    LIST OF VARIABLES
    C7    LIST OF CONSTANT TERMS
    C8    LIST OF OPERATIONS
    C9    LIST OF DIFFERENCES
    C10   ORDERING ON RELEVANT DIFFERENCES
    C11   ORDERING ON REL DIFFS, S9 ON C36
    C19   FORMATS FOR DIFFERENCES


  O.K. MOORE PROBLEMS
    C20   PROBLEM OKMOORE A1
    C21   PROBLEM OKMOORE A2
    C22   PROBLEM OKMOORE A3
    C23   PROBLEM OKMOORE A4
    C24   PROBLEM OKMOORE B1
```

```
    C25   PROBLEM OKMOORE B2
    C26   PROBLEM OKMOORE B3
    C27   PROBLEM OKMOORE B4
    C28   PROBLEM OKMOORE C1
    C29   PROBLEM OKMOORE C2
    C30   PROBLEM OKMOORE C3
    C31   PROBLEM OKMOORE C4
    C32   PROBLEM OKMOORE D1
    C33   PROBLEM OKMOORE D2
    C34   PROBLEM OKMOORE D3
    C35   PROBLEM OKMOORE D4
    C36   PROBLEM OKMOORE ALPHA 1
    C37   PROBLEM OKMOORE ALPHA 2
    C38   PROBLEM OKMOORE ALPHA 3
    C39   APPLY B19 TO B74
    C90   APPLY B23 TO B90
    C91   APPLY B24 TO B91
    C92   APPLY B25 TO B92
    C93   TRANSFORM B93 INTO B8
    C97   TRANSFORM B94 INTO B95
    C98   TRANSFORM B97 INTO B96
    C99   TRANSFORM B99 INTO B98
    C109  HIGHEST DEFINED REGIONAL


DIFFERENCES
    D1    ADD TERMS (EXTRA TERMS IN NBR 2)
    D2    DELETE TERMS (EXTRA TERMS IN NBR 1)
    D3    INCREASE TERMS (MORE OFTEN IN NBR2)
    D4    DECREASE TERMS (MORE OFTEN IN NBR1)
    D5    CHANGE CONNECTIVES (NEITHER -)
    D6    CHANGE SIGN (NBR 1 -, NBR 2 +)
    D7    CHANGE SIGN (NBR 1 +, NBR 2 -)
    D8    CHANGE LOWER SIGN
    D9    CHANGE POSITION
    D10   CHANGE GROUPING A(BC) TO (AB)C
    D11   CHANGE GROUPING (AB)C TO A(BC)
    D12   NBR 1 HAS MULTIPLE NOTS
    D13   NBR 2 HAS MULTIPLE NOTS
    D14   VAR NBR 1 VS EXP NBR 2
    D15   EXP NBR 1 VS VAR NBR 2
    D16   CONSTANT NBR 1 VS EXP NBR 2
    D17   EXP NBR 1 VS CONSTANT NBR 2
    D18   CONSTANT VS CONSTANT
    D19   EXP NBR 1 HAS COMMA, EXP NBR 2 NOT
    D20   EXP NBR 2 HAS COMMA, EXP NBR 1 NOT
    D21   BOTH HAVE COMMA, EXP NBR1 MORE OPER
    D22   BOTH HAVE COMMAA, EXP NBR2 MORE OPR
    D23   BOTH HAVE COMMA, SAME NBR OPERANDS
    D24   TEX NBR 1 VS. EX NBR 2
    D25   EX NBR 1 VS TEX NBR 2
```

EX 1 VS. EX 2, UNMATCHED TERMS
- D30 NONE, NONE
- D31 NONE, SOME
- D32 SOME, NONE
- D33 SOME, SOME
- D34 ALL, ALL
- D39 UNION OF INDEPENDENT SUB DIFFS
- D69 HIGHEST DEFINED REGIONAL

CONSTANTS, GOAL TYPES, FORMS, ETC.

GOAL TYPES
- K1 TRANSFORM OBJECTS GOAL TYPE
- K2 APPLY OPERATOR GOAL TYPE
- K3 REDUCE DIFFERENCE GOAL TYPE

GOAL FORMS
- K11 TRANSFORM GOAL FORM
- K12 APPLY OPERATOR GOAL FORM
- K13 REDUCE DIFFERENCE GOAL FORM

ATTRIBUTE TRANSFER LISTS
- K20 TRANSFER LIST G21, G11, G2, G12
- K21 TRANSFER LIST G1, G11, G2, G12
- K22 TRANSFER LIST G11, G2, G12
- K23 TRANSFER LIST FOR ALL COMPONENTS
- K29 TRANSFER LIST FOR COPYING TEX

CRITERIA FOR LIMITS
- K30 CRITERIA FOR ABSOLUTE NBR OF GOALS
- K31 CRITERIA FOR RELATIVE NBR OF GOALS
- K32 CRITERIA FOR ABSOLUTE EFFORT
- K33 CRITERIA FOR RELATIVE EFFORT
- K34 CRITERIA FOR ABSOLUTE DEPTH
- K35 CRITERIA FOR RELATIVE DEPTH
- K36 CRITERIA FOR ABSOLUTE NBR OF OBJECT
- K37 CRITERIA FOR RELATIVE NBR OF OBJECT
- K38 CRITERIA FOR ABSOLUTE NBR MTH TRIES
- K39 CRITERIA FOR RELATIVE NBR MTH TRIES

METHODS
- K40 MATCH METHOD FOR TRANSFORM GOAL
- K41 TRY OPR METHOD FOR APPLY OPR GOAL
- K42 RELEVANT OPR METHOD FOR REDUCE GOAL
- K43 TRANSFER RESULTS METHOD

MISCELLANEOUS
- K50 BLANK LIST
- K51 TEMPORARY STORAGE
- K52 TEMPORARY STORAGE
- K53 TEMPORARY STORAGE
- K54 TEMPORARY STORAGE
- K56 LIST OF GPS VAR
- K59 GPS TABLE OF CONNECTIONS

OPERATOR EXPRESSION OPERATIONS
- K60 REVERSE (OPR EXP OPERATION)

TASK ENVIRONMENTS
- K70 TE FOR O.K. MOORE LOGIC

TE GENERAL REFERENCE
- K80 REMOTE INFORMATION ABOUT TE
- K81 DIFFERENCE ORDERING
- K82 LIST OF VARIABLES
- K83 GOAL VALUE TYPE
- K84 DSC LIST OF DIFFERENCE FORMATS

MISCELLANEOUS
- K90 NONEXISTANT GOAL SYMBOL
- K91 NONEXISTANT ATTEMPT RECORD
- K92 NONEXISTENT TE
- K97 FORM FOR A70 LIST FOR BINARY OPR
- K98 REFERENCE TREE FOR ABSOLUTE
- K99 REFERENCE TREE FOR INCREMENTAL

VALUES
- K101 GOAL VALUE, LEVEL, DIFF
- K102 GOAL VALUE, MAX - LEVEL, DIFF
- K103 GOAL VALUE, DIFF, LEVEL

```
  IMMEDIATE OPERATORS
    K110  GPS IMMEDIATE OPERATOR LISTS
    K111  GPS IMMEDIATE OPRS FOR R31


  CONTENT TYPES
    K161  OBJECT TEX
    K162  OPERATOR
    K163  EQUIVALENCE LIST
    K179  HIGHEST DEFINED REGIONAL


EXPERIMENTER LISTS
  SET UP LISTS
    L1    LIST OF IDENTIFICATIONS
    L2    LIST FOR OFF TRACE
    L3    TRACE LIST FOR Q=3
    L4    TRACE LIST FOR Q=4
    L5    LIST OF SIGNALS
    L6    LIST OF ATTRIBUTES
    L7    LIST OF ROUTINES FOR SIGNAL TRACE
    L8    A2 MONITOR LIST
    L9    H3 MONITOR LIST
    L10   LIST OF GOALS FOR A PROBLEM
    L11   LIST OF TEX'S FOR A PROBLEM
    L12   LIST OF LISTS OF EQUIVALENT GOALS
    L13   LIST OF EXTERNAL TEXS
    L17   LIST FOR TE MODIFICATION
    L18   LIST OF NAMES
    L19   POPUP LIST OF OBJECT NAMES
    L29   GOAL ATTRIBUTES TO KEEP IN CORE


  PRINTING FORMATS
    L30   FORMAT FOR PRINTING GOAL NAME
    L31   FORMAT FOR TRANSFORM GOAL PRINT
    L32   FORMAT FOR APPLY GOAL PRINT
    L33   FORMAT FOR DIFFERENCE GOAL PRINT
    L40   FORMAT FOR SELECTED TEX
    L41   FORMAT FOR TEX
    L42   FORMAT FOR PRINTING OPERATOR
    L43   FORMAT FOR SAME OBJECT
    L44   FORMAT FOR TOO COMPLEX
    L45   FORMAT FOR GOAL SOLVED
    L46   FORMAT FOR GOAL FAILED
    L47   FORMAT FOR NO GOOD
    L48   FORMAT FOR GOAL SELECTED
    L49   PRINT DERIVATION LIST
    L50   NBR 2 L30 ALTERNATIVE
    L51   NBR 2 L31 ALTERNATIVE (K1)
    L52   NBR 2 L32 ALTERNATIVE (K2)
    L53   NBR 2 L33 ALTERNATIVE (K3)
    L109  HIGHEST DEFINED REGIONAL
```

```
MISSIONARIES AND CANNIBLES TE
    M1    LIST OF OPERATORS
    M2    TABLE OF CONNECTIONS
    M3    IMMEDIATE OPERATOR
    M6    LIST OF VARIABLES
    M7    LIST OF CONSTANTS
    M9    LIST OF DIFFERENCES
    M10   ORDERING OF DIFFERENCES
    M11   LIST OF ADMISSIBILITY TESTS
    M19   TE FOR MISSIONARIES AND CANNIBALS


  OPERATORS
    M30   LM
    M31   LC
    M32   LMC
    M33   LMM
    M34   LCC
    M35   RM
    M36   RC
    M37   RMC
    M38   RMM
    M39   RCC


  DIFFERENCES
    M40   -B ON L
    M41   -B ON R
    M42   -M ON L
    M43   -M ON R
    M44   -C ON L
    M45   -C ON R
    M46   -MC ON L
    M47   -MC ON R
    M48   -MM ON L
    M49   -MM ON R
    M50   -CC ON L
    M51   -CC ON R
    M52   3 MC ON R
    M53   3M   ON R
    M54   3C   ON R
    M57   4 ON R
    M58   5 ON R
    M59   6 ON R


  INADMISSIBILITY SIGNALS
    M60   1 EXTRA C ON L
    M61   1 EXTRA C ON R
    M62   2 EXTRA C ON L
    M63   2 EXTRA C ON R
```

```
  OBJECTS
    M70   L = BMMMCCC, R = -
    M71   L = -, R = BMMMCCC

  PROBLEM
    M80   TRANSFORM M70 INTO M71
    M109  HIGHEST DEFINED REGIONAL


INTEGERS 0 THRU 99
    N0
    N1
    N2
    N3
    N4
    N5
    N6
    N7
    N8
    N9
    N10
    N11
    N12
    N13
    N14
    N15
    N16
    N17
    N18
    N19
    N20
    N21
    N22
    N23
    N24
    N25
    N26
    N27
    N28
    N29
    N30
    N31
    N32
    N33
    N34
    N35
    N36
    N37
    N38
    N39
    N40
```

N41
N42
N43
N44
N45
N46
N47
N48
N49
N50
N51
N52
N53
N54
N55
N56
N57
N58
N59
N60
N61
N62
N63
N64
N65
N66
N67
N68
N69
N70
N71
N72
N73
N74
N75
N76
N77
N78
N79
N80
N81
N82
N83
N84
N85
N86
N87
N88
N89
N90
N91
N92
N93

```
    N94
    N95
    N96
    N97
    N98
    N99
    N209  HIGHEST DEFINED REGIONAL


LOCAL CONTEXT CELLS
    Y1    SIGNAL
    Y2    CURRENT GOAL
    Y3    GOAL TYPE
    Y4    TASK ENVIRONMENT
    Y5    CURRENT METHOD
    Y6    LOCATION OF CURRENT SEGMENT
    Y7    SUPER GOAL
    Y9    MOST RECENT SUBGOAL
    Y10   EQUAL GOAL, IF EXISTS
    Y11   EX 1 (LOC)
    Y12   EX 2 (LOC)
    Y13   TEX 1
    Y14   TEX 2
    Y15   LOC PROGRAM FOR EX 1
    Y16   LOC PROGRAM FOR EX 2
    Y17   DIFFERENCE NET
    Y18   DIFFERENCE TYPE
    Y19   ADDITIONAL LOC PROGRAM (INVERTED)
    Y20   OPERATOR
    Y21   LOC IN LIST OF RELEVANT OPERATOR
    Y22   LIST OF OPERATORS TRIED
    Y23   TEST FOR OPERATOR CONDITION
    Y24   TEST FOR OPERATOR PRODUCT
    Y25   NET OF GOALS
    Y26   FINAL TEX
    Y27   LOC PROGRAM OF FINAL TEX
    Y28   PRIOR SIGNAL IN Y1
    Y29   LIST OF LIMIT CRITERIA TO BE USED


  LIMITS
    Y30   ABSOLUTE NUMBER OF GOALS
    Y31   RELATIVE NUMBER OF GOALS
    Y32   ABSOLUTE EFFORT (BASE)
    Y33   RELATIVE EFFORT (BASE)
    Y34   ABSOLUTE DEPTH
    Y35   RELATIVE DEPTH
    Y36   ABSOLUTE NUMBER OF OBJECTS
    Y37   RELATIVE NUMBER OF OBJECTS
    Y38   ABSOLUTE NUMBER OF METHOD TRIES
    Y39   RELATIVE NUMBER OF METHOD TRIES
```

```
  Y40   TEMPORARY WORKING CELL
  Y41   TEMPORARY WORKING CELL
  Y42   TEMPORARY WORKING CELL
  Y43   TEMPORARY WORKING CELL
  Y44   TEMPORARY WORKING CELL
  Y45   SIGNAL FOR Y11 COPIED
  Y46   SIGNAL FOR Y12 COPIED
  Y47   SIGNAL FOR Y20 COPIED
  Y48   SIGNAL FOR Y84

TASK ENVIRONMENT CELLS
  Y51   LIST OF OPERATORS
  Y52   TABLE OF CONNECTIONS
  Y53   LIST OF IMMEDIATE OPERATORS
  Y54   LIST OF OBJECTS
  Y55   LIST OF EQUIVALENCE LISTS
  Y60   IDENTITY COMPARISON
  Y61   COMMAND
  Y62   SIMILARITY TEST, OBJECT SETS
  Y63   COMPARE OBJECTS
  Y64   COMPARE OPERATORS
  Y65   SEARCH FILTER ON OPR CONDITIONS
  Y66   SEARCH FILTER ON OPR PRODUCT
  Y67   TEX ADMISSIBILITY TEST
  Y68   ADMISSIBILITY TESTS TO BE DONE
  Y69   STANDARDIZATION
  Y70   SIMILARITY TEST, OPERATORS SET
  Y72   ADJUSTMENT FOR EX1 (Q51)
  Y73   ADJUSTMENT FOR EX2 (Q52)


  Y80   NEW TEX
  Y81   LOC PROGRAM OF NEW TEX
  Y82   IDENTICAL TEX
  Y84   LIST OF DIFFERENCE EXPRESSIONS
  Y85   CONTENT TYPE OF NEW OBJECT
  Y86   METHOD LIST
  Y87   PROPOSED GOAL
  Y88   TEMPORARY FOR PRIOR GOAL
  Y89   ATTEMPT RECORD
```

INDIRECT ROUTINES

Y90	EXECUTIVE
Y91	MATCH
Y92	EVALUATE NEW SUBGOAL
Y93	MAKE VARIABLES DISJOINT
Y94	TOP EXECUTIVE
Y95	CONSTRUCT GOAL VALUE
Y96	SELECTION OF NEW GOAL IN R10
Y97	SELECTION OF SUBGOAL
Y98	DESCRIBE MARKED LIST
Y99	SELECT NEXT DIFFERENCE
Y100	EVALUATE EQUIVALENT SUBGOAL
Y110	CURRENT DIFFERENCE EXPRESSION
Y111	TOP GOAL
Y130	COMPLEXITY LIMIT
Y209	HIGHEST DEFINED REGIONAL

EXPERIMENTER CELLS

Z7	AVAILABLE SPACE LIMIT
Z9	ALPHABETIC BLANK
Z20	BLANK CELL
Z21	TEMPORARY WORKING CELL
Z22	TEMPORARY WORKING CELL
Z23	TEMPORARY WORKING CELL
Z24	TEMPORARY WORKING CELL
Z29	CELL FOR SIGNAL INTERPRETER
Z30	CRITERION FOR SPACE LEFT

WORKING CELLS FOR CONVERSION

Z40	HOLDS INTERPRETATION LIST
Z41	HOLDS ORIGINAL HEAD
Z42	HOLDS HEAD OF CURRENT SUBLIST
Z43	LAST LIST CELL OF CURRENT SUBLIST
Z44	HOLDS CURRENT CELL OF INPUT LIST
Z45	HOLDS INPUT WORKING LIST
Z46	PUSHDOWN LIST OF LOC PROGRAM OF Z43
Z47	HOLDS LOC PROGRAM

WORKING CELLS FOR PRINT

Z50	HOLDS REFERENCE COL NBR
Z51	HOLDS LOC IN FORMAT LIST
Z52	HOLDS SIGNAL IF ANY SIGNALS

```
COMMON WORDS
  Z60   GOAL
  Z61   0 (SUBG
  Z62   OAL 0
  Z63   F
  Z64   )
  Z65   (
  Z66   REJEC
  Z67   TED


EXPERIMENTER TE CELLS
  Z80   CONVERT TEX
  Z81   PRINT TEX


INITIAL SET UP AND MONITORING
  Z90   PROGRAM FOR THIS RUN
  Z91   TASK FOR THIS RUN
  Z92   MONITOR SIGNAL
  Z94   MONITOR GOAL EXP PRINT (E24)
  Z95   MONITOR GOAL NAME PRINT (E25)
  Z96   MONITOR IN E2 PRIOR TO R2
  Z97   MONITOR NEW TEX (Q48)
  Z98   MONITOR RECORD ATTEMPT (Q6)
  Z99   MONITOR LIMITS (Q1)
  Z129  HIGHEST DEFINED REGIONAL


CHARACTER SYMBOLS
    A
    B
    C
    D
    E
    F
    G
    I
    K
    L
    M
    P
    Q
    R
    S
    T
    U
    V
    X
    Y
    Z
```

/
$
*
+
-
.
,
(
)

Appendix D

FIGURES

Transform A into B

Method K40

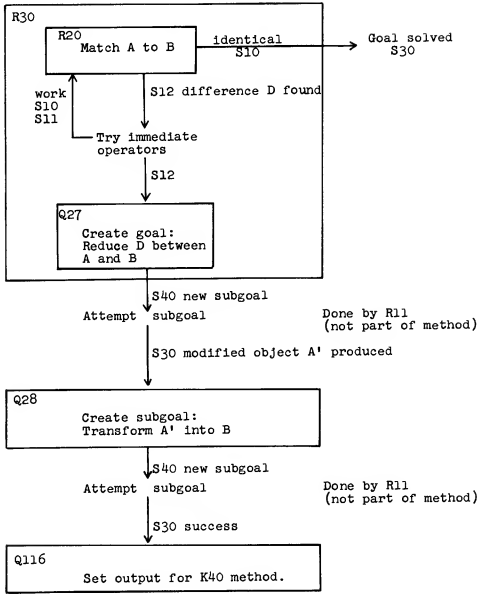


Fig. 1 Rough Flow Diagram for K40 Method.

Reduce D from A to B

Method K42

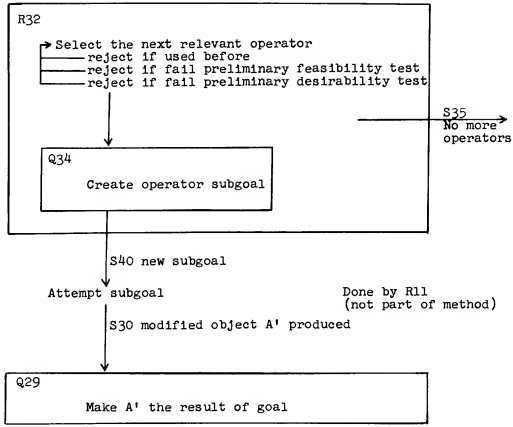


Fig. 2 Rough Flow Diagram for K42 Method.

Apply R to A

Method K41

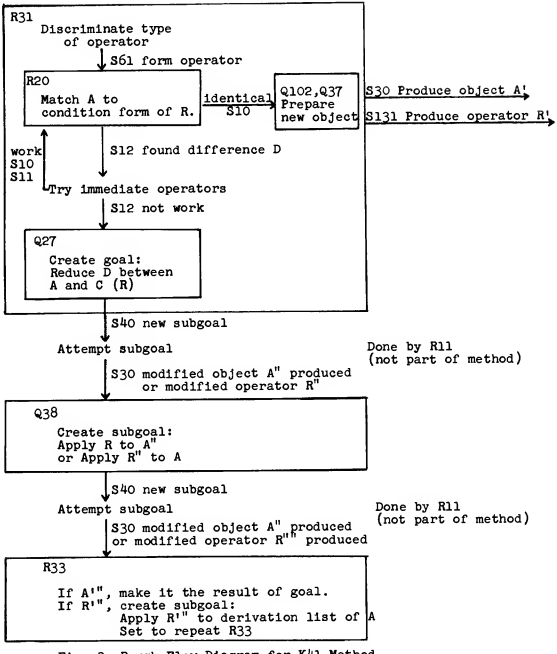


Fig. 3 Rough Flow Diagram for K41 Method.

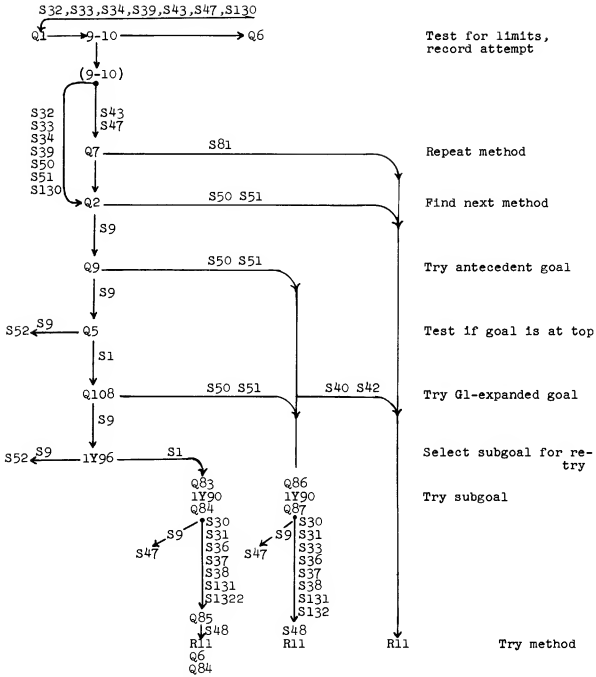


Fig. 4 R10: Problem-Solving Executive.

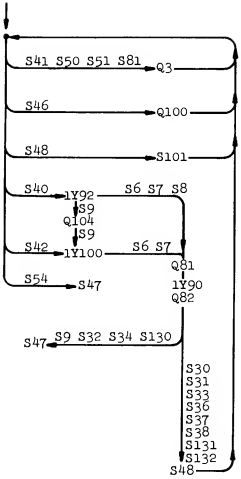


Fig. 5 R11: Executive Method until Fail.

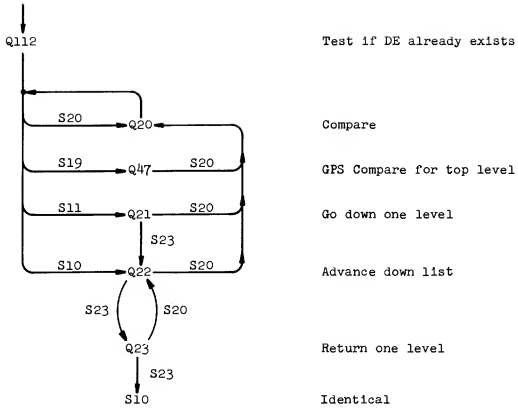
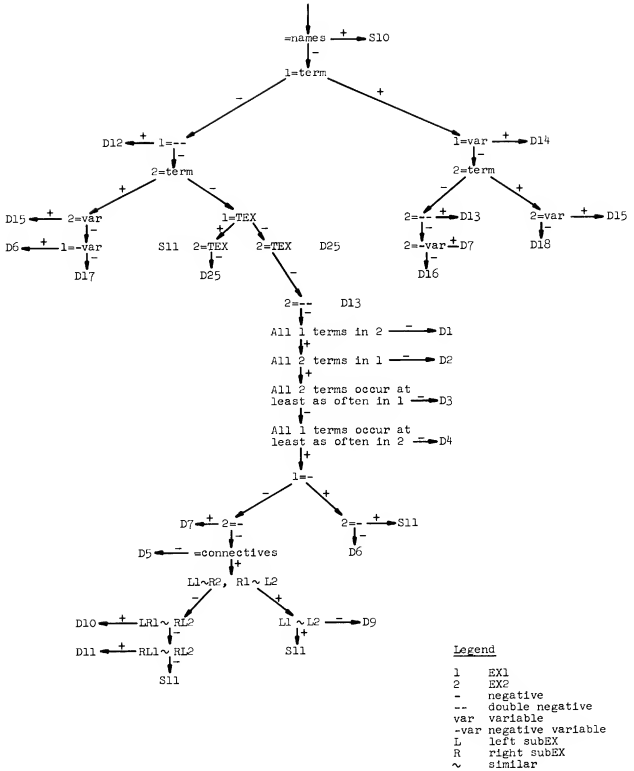


Fig. 6 R20: Match Element by Element, Depth First.



Legend

1	EX1
2	EX2
-	negative
--	double negative
var	variable
-var	negative variable
L	left subEX
R	right subEX
~	similar

Fig. 7 F1: Compare EX1 and EX2.

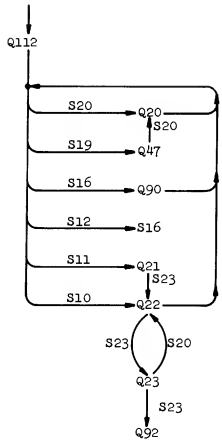
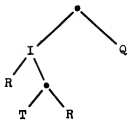


Fig. 8 R21: Match with Single Pass Getting List of Difference Expressions.

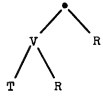
EX1

[RI(T.R)].Q



EX2

(TVR).R




```
Y84/  3725.                                  Cell holding list

3725/ 0                                      List of difference expressions
      9-1                                                    (DE's)
      9-2

9-1/  9-10.                                  DE for lefthand side
9-10/ 0
      A89
      D32                                    D32 = delete from EX1
      A88
      P10                                    P10 = location program "left"
      A87
      9-11   9-11/0    9-12/0                Lists of terms
      A86        R          T
      9-12.      S1         S1
                 T          R
                 S1         S1.
                 R.

9-2/  9-20                                   DE for righthand side

9-20/ 0
      A89
      D34                                    D34 = disjoint terms
      A88
      P11                                    P11 = location program "right"
      A87
      9-21   9-21/0.   9/22/0                Lists of terms
      A86        Q.         R.
```

Fig. 9 Two Matched Expressions.

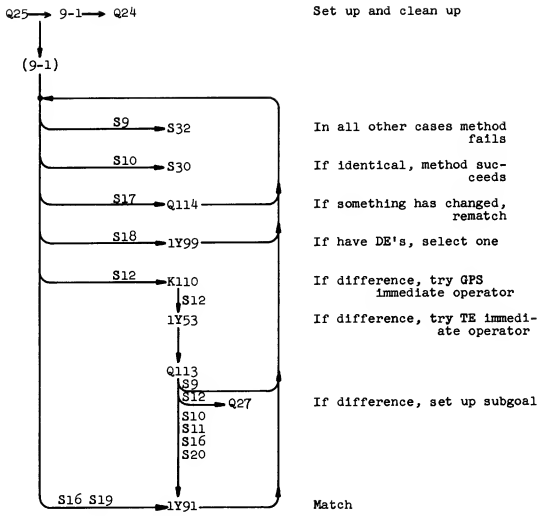
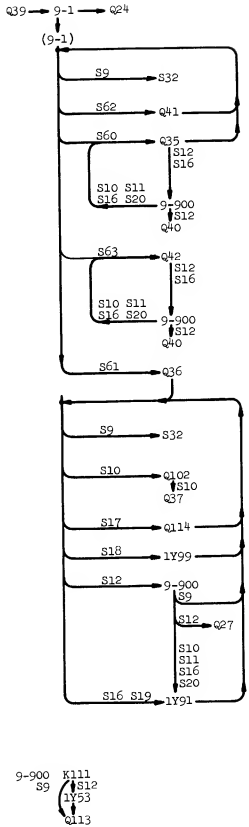


Fig. 10 R30: Match G1 to G2,
If Not Match Produce K3 (Reduce) Subgoal.



In all other cases method fails

Find operator given by expression

Test for operator applicability

Try immediate operators

If still difference, set up subgoal

Try direct operator

Try immediate operators

If still difference, set up subgoal

If a form operator, set up for match

In all other cases method fails

If match, prepare output if product undetermined

Produce product

If something has changed, rematch

If have DE's, select one

If difference, try immediate operators

If still difference, set up subgoal

Match

Try GPS immediate operators

If difference, try TE immediate Operators

Record result

Fig. 11 R31: Try Operator, If Not Work Produce K3 (Reduce) Subgoal.

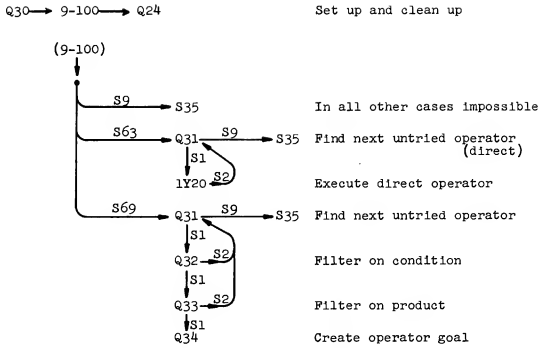


Fig. 12 R32: Find Next Untried Relevant Operator and Produce K2 Goal.

REFERENCES

1. Newell, Allen, and H. A. Simon, The Logic Theory Machine: A Complex Information Processing System, The RAND Corporation, P-868, published also in IRE Transaction on Information Theory, Vol. IT-2, No. 3, September 1956, pp. 61-79.

2. Newell, Allen, H. A. Simon, and J. C. Shaw, Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics, The RAND Corporation, P-951, published also in Proceedings of the 1957 Western Joint Computer Conference, February 1957, pp. 218-230.

3. Newell, Allen, and J. C. Shaw, Programming the Logic Theory Machine, The RAND Corporation, P-954, published also in Proceedings of the 1957 Western Joint Computer Conference, February 1957, pp. 230-240.

4. Newell, Allen, J. C. Shaw, and H. A. Simon, "Preliminary Description of General Problem-Solving Program--I (GPS-1)," CIP Working Paper No. 7, December 1957.

5. Newell, Allen, J. C. Shaw, and H. A. Simon, Report on a General Problem-Solving Program for a Computer, The RAND Corporation, P-1584, also published in Information Processing: Proceedings of the International Conference on Information Processing, UNESCO, June 1959, Paris, 1960, pp. 256-264, and in Computers and Automation, July 1959.

6. Newell, Allen, Some Problems of Basic Organization in Problem-Solving Programs, The RAND Corporation, RM-3283, December 1962.

7. Newell, Allen, J. C. Shaw, and H. A. Simon, The Process of Creative Thinking, The RAND Corporation, P-1320, September 1958.

8. Newell, Allen, and H. A. Simon, The Simulation of Human Thought, The RAND Corporation, P-1734 and RM-2506, also published in Current Trends in Psychological Theory, University of Pittsburgh, 1961, pp. 152-179.

9. Newell, Allen, and H. A. Simon, GPS, A Program that Simulates Human Thought, The RAND Corporation, P-2257, also published in Lernende Automaten, H. Billings (ed.), (Proceedings of a Conference at Karlsruhe, Germany, April 1961), Oldenbourg, Munich, 1961, pp. 109-124.

10. Newell, Allen, and H. A. Simon, Computer Simulation of Human Thought, The RAND Corporation, P-2276, also published in Science, Vol. 134, No. 3495, December 1961, pp. 2011-2017.

11. Newell, Allen, and H. A. Simon, Computer Simulation and Human Thinking and Problem-Solving, The RAND Corporation, P-2312, also published in Management and the Computer Future, M. Greenburger (ed.), Wiley, 1962, pp. 95-131.

12. Newell, Allen, J. C. Shaw, and H. A. Simon, A Variety of Intelligent Learning in a General Problem-Solver, The RAND Corporation, P-1742, also published in Self-Organizing Systems, M. C. Yovits, and S. Cameron (eds.), Pergamon, 1960, pp. 153-189.

13. Simon, H. A., Experiment with the Heuristic Compiler, The RAND Corporation, P-2349, June 1961.

14. Newell, Allen (ed.), Information Processing Language V Manual, The RAND Corporation, P-1897 and P-1918, also published by Prentice-Hall, 1961.

15. Moore, O. K., and S. B. Anderson, "Modern Logic and Tasks for Experiments on Problem-Solving," Journal of Psychology, Vol. 38, 1954, pp. 151-160.